University of Waterloo

Waterloo, Ontario, Canada
N2L 3G1

Faculty of Mathematics
Department of Computer Science
519/885-1211

Telex Number
069-55259

INVOICE

July 16, 1986.

Professor Melvin Klerer,
Polytechnic University,
Dept. of Electrical Engineering
and Computer Science,
333 Jay Street,
Brooklyn, New York 11201
U.S.A.

Dear Professor Klerer:

Thank you for your letter of June 12, 1986.

Enclosed please find our technical report CS-80-49, by J.S.N. Elvey. Please note that the price for this report is $5.00 CDN. If you would make your cheque payable to the Computer Science Dept., University of Waterloo, it would be greatly appreciated.

Thank you for your interest in our department.

Yours truly,

Susan DeAngelis

Susan DeAngelis (Mrs.),
Technical Report Secretary.

/sd

Encl.

received payment $5.00 U.S.
Aug. 5/86 S.D.

Polytechnic University, 333 Jay Street, Brooklyn, New York 11201

JUN 19 1986

DEPARTMENT OF
ELECTRICAL ENGINEERING
AND COMPUTER SCIENCE

Computer Science
718/643-3645

Polytechnic

June 12, 1986

University of Waterloo
Department of Computer Science
Waterloo
Ontario, Canada

Sirs:

I would be grateful for a copy of the report by J.S.N. Elvey,

"Symbolic Computation and
Constructive Mathematics"
CS - 80 - 49

Very truly yours,

[illegible]

Melvin Klerer
Professor of Computer Science

MK/pb

333 Jay Street, Brooklyn, New York 11201 718/643-5000
Route 110, Farmingdale, New York 11735 516/454-5111
456 North Street, White Plains, New York 10605 914/949-2338

SYMBOLIC COMPUTATION
and
CONSTRUCTIVE MATHEMATICS

by

John S.N. ELVEY*

Research Report CS-80-49

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

* Woerd Lodge, FARINGDON, Oxon, UK.

SYMBOLIC COMPUTATION

and

CONSTRUCTIVE MATHEMATICS

by

John S.N. ELVEY

Woerd Lodge,

FARINGDON,

Oxon,UK.

0. Introduction.

1. Historical sketch.Remarks on the rôle of LISP.

2. MACSYMA--a general description.

3. ALTRAN.

4. ANALITIK.

5. CAMAL.

6. FORMAC.

7. REDUCE.

8. SAC-1.

9. SCRATCHPAD.

10.SYMBAL.

11.Special packages.

12.Comments on scope;and,on some fundamental algorithms.

13.Remarks on I/O and simplification.

14.Symbolic analysis(the mathematical use of symbolic computation).

15.Mathematical applications.
(15.2 Implicit functions of one complex variable.15.3.Computational function theory. 15.4.Calculations involving Patil's(and analogous)representation(s).15.5.Summability procedures;tauberian theorems;asymptotic analysis.15.6.Differential calculus in general spaces.15.7.Constructive conformal mapping of general,n-connected domains. 15.8.Branching analysis for nonlinear operator equations.15.9.Function theoretic methods in elasticity.15.10.Approximate solution of ill-posed problems.15.11.Design of switching circuits,using Galois fields.15.12.Stability analysis for(systems of) (non)linear differential equations.15.13.Basic calculations in (algebraic)topology. 15.14.Operational calculus;generalized functions.15.15.Operational methods for partial differential equations.15.16.Topological approximation of fixed points of mappings.15.17.Calculations in singularity theory.15.18.Finite element(and related) methods.Calculations involving bases in Banach spaces.15.20.Some calculations in rigorous statistical mechanics.15.21.Approximate solution of (stochastic)(functional) operator equations.15.22.Miscellaneous topics:(a)calculations in general perturbation theory;(b)calculations in abstract harmonic analysis;(c)calculations in computational geometry;(d)calculations in probability theory and mathematical statistics; (e)calculations in analytic complexity theory.)

16.Remarks on 'efficiency' and comparison.

17.Bibliographical notes,and general conclusions.

18.References.

0. INTRODUCTION

Since the early Nineteen-Sixties,various systems have been developed for symbolic computation.Between them,these systems encompass a wide range of analytical and algebraic calculations,surpassing by far the unaided efforts of human calculators. Naturally,a crucial factor is the quality of the algorithms used;and the versatility of the hardware/software packages has been achieved only gradually—as a result of experiment.Special languages have been designed to allow the I/O to be as 'natural' as possible in some packages;while others seek to extend high-level numerical languages just sufficiently to include the symbolic facilities required.Althoughmany problems must be overcome before symbolic computation systems are as readily available as,say,FORTRAN or ALGOL,recent design improvements and new algorithms have been so successful that a survey of systems,and of their potential use in nontrivial mathematical problems is overdue.Previous reviews in this area(Barton and Fitch(1972a,b)Cohen,Leringe and Sundblad(1976),Hearn(1977),Brown and Hearn (1978),d'Inverno(1978))have omitted some systems altogether,and concentrated on problems in general relativity,quantum electrodynamics and celestial mechanics, where the size of the computations is the main obstacle,the mathematical procedures being fairly straightforward.Moreover,they have not described the structure of the languages,or their mathematical scope,except in the barest outline(though the two papers of Barton and Fitch do contain good description of some of the basic algorithms—as of about 1971).There are also shorter articles by Miola(1976a,b—in Italian),and Sundblat(1976),but these do not alter the position.

The impact of the latest developments in symbolic computation should not be confined to the improved performance of comparatively simple but tedious manipulations.Rather,the increasingly sophisticated facilities constitute both an invitation and a challenge to mathematicians to (re)formulate procedures as constructively as possible,to exploit this powerful aid in calculation.There are signs of some movement in this direction.The group theory system,GROUP,with its associated language,CAYLEY(Cannon(1976))is highly developed;Stoutemyer((1974a,b),(1975), (1977a,b))has adopted an analytical approach to certain problems in optimization, calculus of variations,solution of equations in finite terms,error analysis,and analytical solution of integral equations.Davenport(1979a,b) and Trager(1979), building on earlier work of Risch((1969),(1970))are developing effective methods for the integration of algebraic functions(though most of the mathematics is hidden:the aim is to provide a strong integration facility rather than to apply symbolic computation in a new field).However,the scope for development in this area is enormous;and the level of sophistication now reached makes it important to identify and outline potential fields of application where complicated mathematical constructions may be implemented.A start in this task is made in Section 15.

It may be argued very strongly that the dominant ingredient in all mathematical activity is approximation--identities of all kinds,however important,forming at any time a 'small' subset of special cases within a morass of partially solvable problems.Although this idea is palpable in computational contexts,its validity within realms of mathematics apparently remote from computation is seldom recognized,and many interesting lines of investigation are thereby ignored.For instance,a matter of geat interest is the definition of 'spaces of problems',and the study of possible topologies for such spaces.The use of symbolic computation,in a suitably general way,combined with techniques of complexity theory(especially those based on measures of 'data information')seems likely to produce fundamental results here. (See the conference proceedings edited by Traub(1975,1976),and the monograph by Traub and Wozniakowski(1980)for details of the information-based approach,and for related papers and references on analytic and computational complexity.)

One of the aims of this paper is to show that symbolic computation may be used as a standard resource in mathematical research,bringing within the realms of practical calculation many effective procedures of considerable subtelty which are at present of purely formal interest(as they are too cumbersome to use,except in essentially trivial cases)and prompting the development of such procedures in areas so far almost devoid of constructive content.This aim entails the identification of mathematical schemes where symbolic computation,properly used,could play a basic(often,crucial)rôle.There are several novel elements in the approach adopted in Section 14,and in the problems discussed in Section 15.They are intended to stimulate interest,rather than to arrive at best possible formulations, since it would be premature to seek definitive results at this early stage.

The other principal aim is to give outlines of all the established systems,with accounts of the corresponding languages and of the mathematical processes covered. On this level,it is possible to assess therelative scope of the systems,and to indicate those most suitable for particular types of calculations.Further comparisons may be made on the basis of various attributes related to the writing and running of programs.However,modifications are made almost daily in details of design, and the only reliable way to ascertain the current situation for any of the packages is to consult members of the design groups(the addresses are given in the Bibliography).Even so,there are linguistic peculiarities and limitations in scope which transcend such fine-structural details as may be in a constant state of flux;and it is these macrostuctural characteristics(which largely determine the system(s) best suited to particular 'users')that are of primary concern here. Since relatively few people are familiar with even the basic ideas of symbolic computation,the most sensible course seems to describe one system in comparative detail;after which the somewhat briefer descripitons of other systems will be comprehensible,and adequate for the identification of special features.

It should be clear that 'the optimal choice of systems' is seldom well-defined. There are so many competing requirements that it is most unlikely for any system to be optimal in relation to all of them.Examples of important characteristics are: size of memory;speed of operation;clarity of I/O;ease of practical use;mathematical scope;algorithms used;editing facilities;existence of interactive(as opposed to batch)modes;quality of interface with high-level numerical languages;portability; potential for user-modification;error detection facilities;ease of learning.This is far from being an exhaustive list,but it is apparent that some,at least,of these attributes are incompatible.So the problem of system design is one of skilful compromise—which accounts,in part,for the variety of packages available,matters of emphasis determining their dominant characteristics.

Unfortunately,it is impractical to perform calculations in several stages,on different systems,since the variations in I/O do not allow this,and the use of compilers or preprocessors would be prohibitively complicated.Thus,users must decide once and for all which single system to use in a given calculation.It will become clear that,with some knowledge of the list processing language,LISP(MacCarthy et al.(1959),Berkeley and Bobrow(1964),Weissman(1967),Allen(19),Maurer(1972),LISP Machine Manual(MIT,19))a user can add facilities to some packages without undue effort.Other systems,although possessing a far larger stock of built-in routines, can be modified only with extreme difficulty.This must be borne in mind in any discussion of the comparative merits of systems.Flexibility is especially important in the most sophisticated mathematical applications.

My own interests lie in constructive mathematics,rather than in system design.The preferences I express represent judgements on the mathematical scope,versatility and outward logical coherence of the packages.Questions of relative cost(under near-optimal use),elegance of internal organization or subtelty of editing or debugging facilities,though undoubtedly important,are outside my purview and perhaps deserve an article of their own.Basic information of this kind may be found in the manuals,and in the sporadically issued newsletters and technical reports(see also the review articles referred to above).When all relevant aspects of performance are taken into account,no single system is absolutely the best ;and it is notable that the opinions of designers about their own and rival systems cannot be taken always at face value!However,a tolerable element of competition adds to the charm of the subject.In view of my emphasis on mathematical procedures of high complexity,I shall describe the MIT system,MACSYMA(The Mathlab Group,1968 onwards)in somewhat greater detail than is given for the others .As a self-contained package,MACSYMA offers the most built-in facilities and,by briefly describing these, one gets an idea of the scope attained(and potentially attainable in some other systems,given sufficient user-participation.See Sections 12 and 13 for more remarks about this).

Thus,I do not claim to give an objective ordering of systems(indeed,it is clear that no absolute ordering can be justified).On the other hand,outstanding merits are mentioned and obvious defects are identified.Readers are referred to the design groups on all matters of technical implementation,availability and current states of development.(As examples of attempts at 'semi objective comparison', based on timings,two tables,due to Cohen et al.(1976)and d'Inverno(1978),are reproduced in Section 16).

It is hoped that the outlines given here,though brief,contain all of the essential information that does not depend on the particular implementations of each system,so that potential mathematical users can decide which system(s) to use.It appears that the majority of computer scientists(let alone mathematicians in general)are practically unaware of the existence of symbolic computation.This is especially regrettable because the most fruitful path for research in this area lies in the extension of combined symbolic and numerical facilities.All of the examples considered in Section 15 could—and should—lead ultimately to calculations with a numerical component.This paper should be understood as an outline description of symbolic computation systems,viewed from the perspective of MACSYMA, and emphasizing nontrivial mathematical procedures.

The layout of the paper is as follows.Section 1 contains a historical sketch, giving an idea of the pioneering work in this field,and of the key rôle played by LISP.The outline of MACSYMA is given in Section 2,covering briefly the mathematical scope and certain aspects of the data-handling and evaluation procedures.The more abbreviated outlines,Sections 3 to 10,cover,respectively ALTRAN(Brown et al., (1968-1973));ANALITIK(Glushkov et al.(1971;1978));CAMAL(Fitch(1975));FORMAC(Tobey and Sammet (1967);Knoble and Bahr(1973));REDUCE(Hearn(1973));SAC-1(Collins et al.(1976)); SCRATCHPAD(Griesmer,Jenks and Yun(1975)),and SYMBAL(Engeli(1966-)).Of course,all of these systems have been developed by many different people,at various times;the citations here refer only to those workers associated most closely with particular packages.Section 11 contains short descriptions of certain special-purpose packages,among them,those for relativity,and for finite groups.In Section 12, remarks are made on relative scope(including comments on algorithms that have been developed largely for use in symbolic computation),while a discussion of I/O and problems of simplification is given in Section 13,together with reproductions of the listings and sample I/O of programs for various systems.

The potential use of symbolic computation as an aid in general mathematical research is discussed in Section 14,where the design of an 'environment for investigation' is emphasized.In Section 15,a brief discussion is given of diverse examples(of which many more could have been adduced,but for lack of space).I am developing some costructive routines,on which experimental programs may be based, though this is a longterm project.Hopefully,this article will stimulate others to study analogous problems,the ultimate aim being to accumulate a library of procedures ranging widely over mathematics and its applications.The examples have been chosen deliberately to include many different mathematical fields,as this adds point to my claim that symbolic computation can be used effectively as a standard research tool.

Of particular importance is the attempt to incorporate fundamental results in 'operational form'.This is a significant extension of the idea of 'side relations', which are used mainly as aids to simplification.It requires that modifications of the system may be made by a user fairly easily(though there is also scope for this approach in large 'fixed' systems).The aim is to allow various results(in the form of 'theorems')to be applied—for instance,in the construction of error estimates for approximation schemes—the choice of these results being dictated by the problem under consideration.Several systems have the potential for extensive use of this strategy,which largely characterizes 'mathematical',as distinct from 'numerical-computational',analysis.Indeed,to emphasize this distinction,I introduce the term symbolic analysis to describe the fullest simulation by computer of mathematical activity(in all fields).

The variety of the examples is so great that space(and competence)allow only the barest indications of effective procedures to be given;but these are complemented with references and suggestions for more detailed schemes.In fact,many of the examples were suggested by the existence of potentially constructive but totally impractical treatments,to which references are given.With the help of symbolic computing,some of these methods may be converted into powerful approximation procedures.What is essential is the effective reduction of each scheme to a set of (coupled)subroutines that are covered(at least,in principle)by existing symbolic computation packages—or else,have a good chance of being covered in the near future.The practicality of these schemes can be judged only after programs have been developed for them;but the task of furnishing potentially suitable mathematical algorithms is in itself of great importance,as many numerical analysts have recognized in recent years(especially in the solution of partial differential equations—see,e.g.,Wendland(1979)).Naturally,the existing libraries of subroutines make a substantial contribution(they provide many of the basic facilities upon which more elaborate algorithms depend);but they hardly touch on any of the examples considered here,being concerned mainly with the efficient solution of problems involving linear algebra and ordinary differential equations,using well known methods.

The object of Section 15 is to draw the attention of mathematicians to a plethora of fascinating possibilities for the use of symbolic computation.Up to now,there have been remarkable advances in system/language design,and in basic algebraic/ analytical algorithms;moreover,the computational results in several areas of theoretical physics and applied mathematics have been spectacular.However,the mathematical scope of all this activity has been extremely limited,and it is this scope that the present article aims to enlarge.

In Section 16,observations are collected on aspects of symbolic computation lying on the boundary between computer science and mathematics.These include efficiency and algorithm design,and some possible criteria for (objective)comparison of systems.Finally,Section 17 is devoted to concluding remarks,bibliographical notes and comments on some of the items given in the list of references.

Before concluding this introduction,I should explain what is meant,in this paper, by constructive mathematics,since there are many interpretations of this term.The essential characteristics of constructiveness,in all of the following considerations,is that(in principle)effective implementations of procedures(possibly,involving many different types of approximations)are realizable—with associated(intermediate and cumulative)error estimates.This requirement is in line with that of 'approximate realizability' of abstract procedures,in forms amenable to symbolic analysis.Although it is demonstrated only occasionally that particular error estimates are valid,no procedure is considered for which the potential determination of suitable estimates is not not obviously feasible(the crucial problem being to derive them in forms optimal for the investigations at hand).This pragmatic usage of the word 'constructive' contrasts sharply with the more formal,and rigid,requirements of Intuitionism(see,e.g.,Beth(1968)),and,of 'constructive analysis',in the sense of Bishop(1967)—see,also,Bridges(1979).Yet another variant of constructivism stems from the systems of 'constructive logic' developed(mainly)by Soviet mathematicians(see,e.g.,Idel'son(1964),Sanin(1964),Phan(1970)).All of these approaches to constructivity have attractive features,but they are(in spite of their apparently practical basic aims)highly theoretical and restrictive.Even though there are cases where a suitable constructive formulation of a problem yields error estimates,while the classical version does not do so,the use of such formal constructive methods in symbolic computation must await their systematization,and simplification;as well as a broadening of the range of procedures known to admit strictly constructive realizations.

1. HISTORICAL SKETCH.REMARKS ON THE ROLE OF LISP

1.1 The advantages of symbolic computation.

Before indicating how the subject has developed,I offer some arguments to justify the use of symbolic computing.For the majority of computer users,'computing' still means the mechanical performance of numerical procedures,and even those people who have seen systems now in use tend to think of them as largely irrelevant to practical calculation.However,this view is quite mistaken.It is true that,in many cases,the aim of a calculation is to obtain numerical estimates for specified properties(of a model of some phenomenon);but the point at issue is at what stage in the calculation numerical values are assigned to the variables.By delaying as far as possible the numerical phase,one can avoid the needless repetition of a chain of computations,where each repetition is required purely to cover changes in the numerical values of a few basic variables.If an analytic form of the results can be obtained then the subsequent use of a succession of numerical values for the original data achieves the basic aim and reduces the chances of round-off error or machine error.Moreover,from the form of the analytical results,predictions may be made which could never be made on the basis of purely numerical output.This is so especially where one is dealing with functions dependent on parameters that do not enter directly into the main calculation.The form of this parameter dependence cannot be deduced from numerical values alone,in a reliable way.
Another reason for using symbolic computing ,even in a problem where the conditions just adumbrated do not hold,is that it may be possible to circumvent the instabilities inherent in the numerical calculation(if this is done from scratch).For instance,it may be that the analytical form of some integral is known,but that, when the corresponding definite integral is required,between specified limits, singularities occur.Many similar examples could be given.

Next,there is the obvious use of symbolic computing to do calculations that would take too long if done 'by hand'(typical calculations in relativity are of this kind—see,e.g.,d'Inverno(1978)).For an account of current activity in applied mathematics and theoretical physics,see Fitch(1979).For an elaboration of the issues raised here,see Hearn(1971).

Lastly,there is the use of symbolic computation as a mathematical tool—the preoccupation of this paper.The main arguments for,and problems in,this area are discussed in Section 14,and illustrated in Section 15;but the bias of the paper is strongly towards this mode of computation,and this colours the system outlines, since it is the mathematical facilities,rather than the problems or limitations in their implementations,that are the major concern.It is not possible to combine these two points of view in an acceptable way;nor is it necessary to do so.

1.2 Historical sketch.

(The following outline is based mainly on the book of Sammet(1969))

The first recorded attempts to write programs for symbolic computation appear to be those of Karimanian(1953) and Nolan(1953),who wrote programs for symbolic differentiation.Evidently,they did not realise the implications of their work,as there was no follow-up or extension,and the subject was forgotten until many years later.The first full system was FORMAC-1,an extension of FORTRAN,designed by Sammet and Tobey(1962-64);a later version(1967)was PL-1 based.Emphasis was placed on simplicity of use,rather than scope or power.The arithmetic was rational(though the interface with FORTRAN did allow floating point arithmetic on IBM 7090 models). The FORMAC/FORTRAN manual appeared in 1965,and the FORMAC/PL-1 version in 1967.

As an example of the use of FORMAC/PL-1,consider the solution of the system of coupled differential equations $dy_i/dx = f_i(x,y_1,\ldots,y_n)$,all y_i being functions of x,with $y_i(0) = y_{i0}$ given constants,and each f_i a polynomial in its arguments, for $i = 1,\ldots,n$.(See the FORMAC-73 manual for a detailed treatment).In spite of its apparent difficulty,all that is needed to solve this problem is substitution, and integration of polynomials in one indeterminate,x.For the program,one specifies the number of equations,the proposed number of iterations,the initial values, y_{i0},and the coefficients of the polynomials,f_i.If one puts,initially,$y_i = y_{i0}$, on the right side of the given equations,integrates the $f_i(x,y_{10},\ldots,y_{n0})$,as polynomials in x,to obtain new polynomials,$y_i^{(1)}(x)$;after which the y_{i0} are replaced(on the right side of the equations) by the $y_i^{(1)}(x)$—and so on,until the k th iterate,$y^{(k)}(x)$ is obtained,k having been specified.

The assignment of values,expressions,etc.,in FORMAC is governed by the LET operator —e.g.,LET X = A + B.Similarly,the FORTRAN READ and WRITE operators govern FORMAC I/O.A good general survey of early work in algebraic manipulation(with some stress on FORMAC)is Sammet(1967).An essential feature of almost all symbolic computation systems is the reduction of the data to an encoding or mapping of symbols into internal representation of integers,or of lists of integers(coefficients,exponents, etc.).It is readily seen that most algebraic/analytical data may be transformed into lists(or lists of lists,...),and that operations on these lists mirror the more familiar mathematical operations.(Basically,it is a matter of replacing all infix operators by prefix operators).This is the main reason for the great power of LISP in the design of symbolic computation systems.

A crucial problem for any system is that of simplification.In the early systems, this was very rudimentary,and current systems are far more sophisticated.However, most of the fundamental questions arising in the design of effective symbolic computation systems were formulated and studied in connection with FORMAC.In this sense,FORMAC is of importance(and its latest versions are competitive in many kinds of calculations).A problems closely related to that of simplification is to control the growth in intermediate expressions formed during a calculation.There are various facilities to reduce the size of expressions,by cancellation,collecting similar terms and imposing side-relations(such as trigonometrical identities). In CAMAL,a use-count facility ensures that an expression no longer required is 'unset'(i.e.,the store allotted to it is cleared).REDUCE,MACSYMA and SCRATCHPAD are based on LISP,so 'garbage collection' is incorporated automatically.More detailed remarks on these matters will be found in discussions of individual systems,and in Sections 12 and 13.

The next system of historical importance to be implemented was MATHLAB,the forerunner of MACSYMA(See,e.g.,Manove et al.(1968)).MATHLAB was an experimental,interactive(on-line)system written in LISP,and including a facility for rational function integration.A mathematically questionable,but linguistically interesting,example of the use of MATHLAB,illustrating some powerful,basic routines is as follows.The aim is to choose a parameter,h,such that the expectation of the random variable,p(u;h)—where u has probability density f(u;x,y)—is maximal for x = y. The program given seems to put x = y before evaluating the x-derivative and equating this to 0;and the second derivative is not considered at all.Nevertheless,the key instruction in the program is of great interest,foreshadowing as it does some of the most useful facilities in the more recent systems.The key line is:

```
'SOLVE('SUB(Y,X,'DERIV(DEFINT('P1*'F(U),U,Y-S*Y,X)+'DEFINT('P2*'F(U),U,X,Y+S*Y),
                                                                        X)) = 0,H) S
```

It is assumed that the density,f,has support [y-sy,y+sy] ,and that p has distinct forms for $u<x$ and $u>x$,these forms being denoted by p_1 and p_2.All of this is transcribed into upper case for programming purposes.This single line of program directs the system to solve for h the equation obtained by differentiating the integral(from y-sy to y+sy) of the product,fp,as a function of u,then putting x=y and setting the final,symbolic result equal to 0.Notice that many operations are compressed into one line of program;so that,if such procedures are combined with various loop constructions,the subsequent demands on storage are likely to be great—and easily overlooked!

Another early system(probably the first one having a strong mathematical orientation)was FLAP,written in LISP for use in a U.S.Naval research laboratory.This had 'atoms', 'lists of expressions', expressions of the form 'A = B' ,where A and B are expressions,and 'functional expressions' of the form $f(x_1,\dots,x_n)$.There were facilities for multilinear and exterior algebra,and the system was used to study integral transforms,tranformation properties of families of matrices,and related matters of use in solving partial differential equations.

SYMBOLIC MATHEMATICAL LABORATORY was designed as a PhD project by W.A.Martin(MIT, 1967).The coding was in LISP.An unusual feature was the possibility of using a 'light pen',pointing at a display screen,to give extra instructions during a calculation.(This facility is used also in ANALITIK:see Section 3).A typical application was to the perturbative solution of nonlinear ordinary differential equations such as $\ddot{x}+\omega^2x = ex^3$, with $x(s) = \Sigma e^k x_k(s)$,and $t(s) = \Sigma e^j t_j(s)$,the dots denoting differentiation in t.Similar applications(in celestial mechanics)were made using CAMAL,which was designed mainly to solve such problems.REDUCE,on the other hand,was developed to handle calculations in high energy physics(e.g.,those involving 'Dirac matrices' and 'Feynman diagrams').It is of interest to trace the influence of these primary aims in the present(general-purpose)CAMAL and REDUCE systems.

The following list includes early systems for symbolic or numerical/symbolic computing.Most of these systems have disappeared,but some of their features have been incorporated in the latest packages.

ALGY(Bernick et al.(1961),on Philco 2000):elementary factorization;simplification of products of trigonometrical functions.

FORMAC(Sammet and Tobey(1962-64)--See Section 6 for the current version.

MATHLAB(MIT Artificial Intelligence Group.See Bobrow(1968)pp 86-97)

ALTRAN(Brown,Bell Labs.,(1961-66)--This initial version was essentially a set of call routines for the algorithms of ALPAK,written in FORTRAN.(See Section 3).

FLAP(U.S.Naval Laboratory).Written in LISP for study of partial differential equations.

MAGIC PAPER(This system used light-pens,push-buttons and 'scrolls'(similar to files).

SYMBOLIC MATHEMATICAL LABORATORY(Martin(1967))--Allowed rational function integration,and the use of a light-pen.

FORTRAN IV;FORMULA ALGOL;NELIAC(Navy electronics system--created its own compilers);MAD(Michigan Algorithm Decoder--Graham(1959));JOSS(an early on-line system); BASIC(Kemeny et al.(1965);MIRFAC(Gawlik,1967.See Gawlik(1963)for background); KLERER-MAY(Columbia University,1963.See Klerer(1964,1965));CPS(Conversational Programming System,on-line,PL/1-like);MAP,AMTRAN,CULLER-FRIED,LINCOLN RECKONER (all of these systems,developed,in part for NASA,are compared in 'The Status of Systems for On-line Mathematical Assistance',Proceedings of the 22nd ACM National Conference,1967);MADCAP(See Goodman(1962));COLASL(See Balke(1962):extensive choice of symbols,and combination of geometrical shapes using 'superposition').

Further remarks on most of these systems may be found in the book by Sammet(1969).

1.3 The Rôle of LISP

There are,essentially,two levels at which a new symbolic computation system may make use of existing computer languages,namely,for I/O and for internal organization.Although the symbolic facilities reqire some new constructional features, it has been possible,in certain systems,to make practically no changes in the I/O language.ALTRAN,FORMAC and SAC-1 are in this category(though SAC-1 is hardly to be regarded as a language at all,being,rather,a set of FORTRAN calls for subroutines.See Section 8).These systems are extensions of FORTRAN or PL/1.Similarly, ANALITIK,MACSYMA,REDUCE and SYMBAL have I/O languages based on ALGOL,and the CAMAL I/O language is based on the original 'Autocode'.However,the innovative content of the ALGOL-based languages is far greater than for those modelled on FORTRAN.

In one obvious respect direct extensions have an advantage:they are easy to learn for anyone familiar with the basic numerical language.Against this one must set their relative lack of optimality in handling purely symbolic expressions(in comparison with languages designed specially for this purpose).In this respect, SCRATCHPAD and ANALITIK are exceptional:each was designed more or less from scratch,with minimal reference to existing languages,in relation to basic design. Moreover,themost fundamental parts of ANALITIK are implemented in <u>hardware</u>.

Nevertheless,most of the I/O languages for symbolic computation have been influenced markedly by the established high-level numerical languages--both as to syntax and procedures.On the other hand,in the matter of internal implementation of these procedures,there is less divergence than is found at the I/O stages:either there is no distinct internal language(apart from the machine language for the computer in use),as in ALTRAN,FORMAC and SAC-1;or else LISP is used.The fundamental suitability of LISP for symbolic computation was touched on in Section 1.The data, coded into lists(or,lists of lists,...)may be manipulated in this form until a 'result' is retrieved,by means of a two-way translator between LISP and the I/O languages.For this reason,it is not necessary to know LISP in order to use a system (except for certain specified facilities in REDUCE),but without some acquaintance with LISP,some procedures may seem unmotivated.Therefore,a few remarks on the structure of LISP will be made now.Detailed treatments are given in the books referred to in Section 0.

LISP,originally designed by J.McCarthy and co-workers at MIT in 1959,is based on a scheme for the representation of partial-recursive functions of symbolic expressions(see,e.g.,Davis(1958)).The difficulties of learning the input format(for example,in balancing large numbers of pairs of brackets)have been overcome,largely, in modern systems.LISP is intended for problems where list-processing,recursion and symbol manipulation are all present,and are the dominant operations.Most modern LISP systems are interactive.The crucial properties of LISP(aside from its natural relation to suitable representations of mathematical expressions,and operations on them)are:its machine independence;and its built-in capability for consistent self-extension(which allows designers to modify or extend a system written in LISP without completely re-designing it).
The variant of LISP on which most symbolic computation systems are based is known as LISP 1.5(LISP 2 having taken too long to become available!).The principal characters in LISP are upper case Latin letter,digits 0 to 9 and brackets ( ).In LISP,there are operators(or functions)and operands(the lists).A program is, roughly,a collection of functions,interrelated as subroutines are in other languages.Programs are activated by invoking one function,which then activates the rest.The basic elements are atoms(=identifiers,or numbers)and S-expressions(= symbolic expressions,or lists),an empty S-expression being denoted by ( )or NIL. (Operations on S-expressions are expressed in a meta-language,as so-called M-expressions--but these are largely ignored in modern systems).The basic functions, operating on (non-atomic)S-expressions are: car(selects the first item in a list); cdr(selects all of the list except the first item),and cons(joins its two argument lists to form a compound list).Other important functions are:
eq(tests for equality between two atomic symbols),and atom(a predicate which returns 'True'when applied to any atom,and'False',otherwise).In cases of inherent ambiguity,a basic LISP construction uses the 'Church lambda symbol',e.g., for assigning argument values to variables.For instance,the notation f(x,y)(a,b) is unclear to a machine.This is resolved as follows: $\lambda((x,y);f(x,y)(a,b))=f(a,b)$; whereas, $\lambda((y,x);f(x,y)(a,b))=f(b,a)$.Obviously,this construction may be used for arbitrarily complicated functions of any finite number of variables.The definition of recursive functions in LISP is achieved by using the operators define and label,the first of which assigns a name,while the second allows that name to be itself written into the definition(thus completing the recursive structure). This is very important for the mathematical applications.

Conditional statements like: IF p_1 THEN k_1 ELSE IFp_2THEN k_2 ELSE ...IF p_n THEN k_n are rendered in LISP as $[p_1 \rightarrow k_1; p_2 \rightarrow k_2; \ldots p_n \rightarrow k_n]$.Here, the k_i are expressions and each p_i has the value T or NIL(= True or False).Thus,the value of the compound statement just given is k_j,where j is the least subscript for which p_i is true(and it is undefined if no p_i is true).

To illustrate the power of LISP in defining functions that would be extremely hard(if not impossible)to define in the numerical languages,and ,at the same time,to show some basic functions in use,the following construction is given;it aims to form a function which selects the first atomic symbol in its argument list,say,x.Note that it is not merely the first item in the list that is required--that is obtained by applying the car operator.A heuristic definition of the function,say,g,is as follows: $g[x] = [\text{atom}[x] \rightarrow x; T \rightarrow g[\underline{\text{car}}[x]]]$ .This can be made respectable in terms of recursion. As another example of a function whose definition would be almost impossible in numerically oriented languages,consider the problem of constructing a function,say,h,satisfying the conditions: $\underline{\text{cdr}}[h[y]] = \underline{\text{cdr}}[y]$ $(\forall y)$,and $h[y] = y$ if $\underline{\text{car}}[y] \not\leq A$~~,~~—where A is given.

Claim:if B is any expression distinct from A,then h may be defined by: $h[y] = [\underline{\text{eq}}[\underline{\text{car}}[y];A] \rightarrow \underline{\text{cons}}[B;\underline{\text{cdr}}[y]];\ T \rightarrow y]$.The reader is invited to verify the validity of this definition.

In summary:the main features of LISP which are important for the design of symbolic computation systems are (i)that lists are the natural I/O for LISP(and (operations on)lists can mirror(operations on)mathematical expressions);(ii)LISP is designed to work efficiently with recursively defined functions,and many basic processes in mathematics use such functions;(iii)LISP is machine-independent(so a system written in LISP should be equally implementable on all machines);(iv) LISP is capable of consistent extension(so systems can be modified relatively easily);(v)LISP has a built-in 'garbage collector'(as soon as all the free space is exhausted,this facility removes all 'inactive',i.e.,unused,lists--the garbage --from the store,and so creates space for new lists to be used).The possibility of comparatively simple extension is of great importance,since it is very hard to forecast how a particular algorithm will 'fit into'the existing structure of the system,at any time.By concentrating on absolute fundamentals,LISP is flexible, though the price of this flexibility is clumsiness.One might ask whether a system written in machine language would be superior to any other form.Moses has estimated that such a machine-based language would operate at about twice the speed of LISP;but the programming resources need$\overset{ed}{\wedge}$to implement nontrivial algorithms in the machine language would be so great that mathematical development of the system would be extremely slow.On balance,therefore,LISP constitutes an acceptable compromise solution to the problem of language design for internal organization of calculations.

2. MACSYMA —A General Description.

MACSYMA is the largest and most versatile of all self-contained systems now available.It is self-contained in the sense that all of the facilities described in the manual are permanently included as optional routines,rather than as potential facilities that could be used only if the system were modified suitably. For some packages,modification is fairly easy,and the central core of fixed procedures is small,but for MACSYMA, the opposite is true.Nevertheless,if one is going to use just one system,without frequent modification,then a wider range of problems can be solved with MACSYMA than with any other system.MACSYMA is an enormous program(written in LISP,but with a parser allowing quasi-mathematical input). The I/O language is similar to ALGOL 60.The manual,which is in its ninth version, is quite comprehensive,each procedure being documented both syntactically and for applications.The list of procedures is expanding all the time,and includes some very sophisticated mathematical algorithms.For instance,the integration capability is particularly strong.It is in two parts;the first based on pattern matching and table-look-up,and the second on a decision procedure due to Risch(1969),about which more will be said in Section 12.Other unusual facilities include a LIMIT operation,which can evaluate several types of limits(e.g.,by applying l'Hôpital's rule,or by using Taylor series),a routine for handling Laplace transforms and their inverses(linked to the integration facility),determination of Taylor and Laurantseries,and solution of certain types of differential and integral equations. MACSYMA is an interactive system.Its output can be two-dimensional,which is important,since user-reaction to the output at any stage determines the course of a calculation.Apart from the detailed mathematical schemes on which its procedures are based,MACSYMA has structural features which help to makethis diversity of facilities possible.The evolution of the structure of MACSYMA is described in a paper by White(1977).

One of the disadvantages of MACSYMA is its enormity,and consequent lack of portability.Up to now,the only implementations are at MIT(on DEC PDP-10,and DEC 20's, with ITS operating system;and on Honeywell 6180 computers,with Multics operating system)and at Berkeley,where a smaller version of MACSYMA has a direct FORTRAN interface--which is useful when numerical computations are involved.However,users do have access to MACSYMA via the ARPA Net;all that is necessary is to know the local ' logging in' convention for the relevant MIT machine.During an interactive session,the system responds to the ALGOL-60 type user input with its own style of two-dimensional output.A detailed manual on the use of the ITS system and a Primer for beginners are among the other documents describing MACSYMA.The system is designed and maintained collectively by the MATHLAB Group,a constantly changing collection of people whose closest parallel is,perhaps,Bourbaki!However, there are some permanent members who have co-ordinated all of the design effort during the past ten years or so--especially,J.Moses,whose improved integration routine('SIN',thesis,MIT,1967)and other fundamental work formed a starting point for the systematic development of MACSYMA.

In the manual,MACSYMA is described as having approximately 221,000 words of compiled code(on a DEC-10,with 36-bit words).No worthwhile upper bound can be given on the size of the system:as the hardware becomes cheaper and smaller,the possibility of designing systems of ever larger capacity will present itself.From this point of view,the algorithms envisaged in Section 15 are certainly feasible.

Although the manual and other documents are essential for a full grasp of the system,it is worthwhile here to sketch the layout,mode of functioning and scope, since,at the basic level,systems have much in common.After this ,it will be unnecessary to repeat certain details in the descriptions of other systems.

2.2 Internal representation of expressions.

Any expression read by MACSYMA is lexically scanned,parsed and then stored in a general LISP internal form(of which there are several);i.e.,any nonatomic expression is represented as a LISP list,whose first element is the main operator of the expression,the remaining elements being operands.For instance,the input expression 2 * X + 3/4 has the LISP representation (PLUS (RAT 3 4) (TIMES 2 X)); while,more generally,the function(say) F(X) - log X becomes (PLUS (F X) (TIMES -1 (LOG X))). Any function used in MACSYMA has an analogous LISP representation(and the same thing applies in all LISP-based systems).

A form of representation for ratios of polynomials,known as CRE(Canonical Rational Expression)is of importance.In this,the variables in use are ordered from 'main' to 'least main',for any given calculation,and for each expression occurring in it. Polynomials have a recursive list representation;e.g., $3x^2 - 1$ (with MACSYMA form 3 * X ^2 - 1)becomes (X 2 3 0 -1),in CRE.Here,X is the main variable; its power is 2 and its coefficient is 3.The constant term(with exponent 0) has coefficient -1.On the other hand,the MACSYMA expression 2 * X * Y + X - 3 has two possible CRE.If X is the main variable,it is (X 1(Y 1 2 0 1) 0 -3)); whereas,if Y is the main variable,it becomes (Y 1 (X 1 2) 0 (X 1 1 0 -3)). In a CRE list,the variables need not be atomic.Another point worth noting—for all systems—is that it is often useful to regard an expression as a polynomial in several 'variables' any distinct entities that it is convenient to treat as 'independent'.For example,in this way,the expression $\sin x + \cos^3(x + 7) + 2\log x$ would become $u + v^3 + 2w$.

The general CRE represents a ratio of two mutually coprime polynomials(with positive 'denominator').Internally,this is a list comprizing the variable ordering list,followed by the numerator and denominator in list form.The variable ordering may be imposed,but if this is not done,then MACSYMA adopts alphabetical ordering. In all of these considerations,the'coefficients' in polynomials may themselves be polynomials,in less main variables.

There is also an 'Extended CRE', to represent Taylor series and more general expressions, in which the exponents may be arbitrary(positive or negative)rational numbers, the corresponding 'coefficients' being arbitrary rational-exponent expressions in the remaining variables. For trigonometrical series, a 'Poisson form' is used by some program packages.

2.3 Some basic operations for input and evaluation.

The usual 'precedence laws for field operations' are imposed automatically in algebraic computations. General MACSYMA expressions consist of numbers, variables, functions calls, and operators(or functions). After the coding into an appropriate internal form, values are assigned to names(which may be subscripted). Functions which produce definite values when acting on evaluated arguments are said to be of verb type; while those which simply return the formal function(with its evaluated argument(s))are of noun type. There is a facility('DECLARE')for converting verb functions into noun functions. Other facilities allow the suppression of evaluation or the performance of extra evaluation(which offers an alternative method of converting 'verbs' into 'nouns'--and vice versa)and the consistent introduction of functions which themselves involve (other)functions.

The general evaluation procedure has the syntax EV(expression,arg_1,...,arg_n),where the subscripted arguments,arg_j,constitue an 'environment for evaluation'--in the sense that they prescribe the basic evaluation processes that will be applied invariably to expressions in the system. The possibilities for the arg_j are wide. For instance: simplification according to specified rules; suppression of evaluation; extra evaluation; repeated evaluation until no change occurs; expansion of products, etc.(possibly setting bounds on the maximum and minimum exponents to be retained); automatic differentiation relative to specified variables(whenever an expression contains a differentiation operation); evalutation of all numerically valued functions, optionally in floating point rather than rational form, and, evaluation of all Boolean expressions--predicates--to give 'T' or 'NIL'. Later on(in Section 14) the question of constructing an 'environment for investigation' will be considered. The arg_j are picked out by the system from left to right, regardless of any ordering that may have been imposed on them, so care is required to avoid inconsistencies. The arg_j may be set either 'locally'(over a segment of a program), or else, 'globally'(over the entire program). However, settings may be altered at any time, for the remainder of the program. Another, crucial facility, involves the substitution of specified expressions into(subexpressions of)the main expression under consideration.

With this background,the process of evaluation in MACSYMA may be described,briefly, as follows.First,the environment for evaluation is specified.Next,all variables in the expression to be evaluated are put into a form suitable for the action of 'EV'.After this,all substitutions indicated in the expression are performed.Then, the expression is subjected to all operations in the environment (unless one of the arg_j suppressed evaluation);and,lastly,if any of the arg_j required extra evaluation,then this is performed repeatedly,until no change is registered in the result.This result is then ready for use in the next stage of the computation.Obviously,this description masks much complicated activity,but it does cover the essential steps.Moreover,although systems may differ considerably in their detailed organization,and in the 'internal languages'used,their procedures for reading in data and evaluating expressions have much in common on a basic level.Thus,the remarks made here apply,in principle,to all systems;so it will be unnecessary to discuss evaluation in the outlines of other systems,except where very unusual features are involved.

2.4. Dissection and simplification of expressions.

A basic problem in symbolic computation is to control the growth of expressions generated in the course of a cacluation.For instance,in the evaluation of determinants,the expressions generated in a straightforward expansion are likely to cause overflow,unless measures are taken to eliminate redundant subexpressions at intermediate stages;and the 'answer',in its simplest form,may consist of a single term.In view of this difficulty,all systems have facilities aimed at maximizing the available storage space and minimizing the size of expressions(at any time). Here again,the principles involved are similar for all systems,but special features will be mentioned for each system described.To some extent,dissection is a necessary precursor to simplification since,in large expressions,it is rarely possible to apply simplification rules 'globally';instead,collections of terms having specified similarities are identified,after which an appropriate form of simplification can be applied to those terms—this procedure being repeated for each collection of terms identified,until maximal simplification is achieved (within the limits of possibility for the system).

There are basic rules(constituting part of the environment of evaluation)which are applied automatically.These include such things as cancellation,imposition of side relations,and use of the relation $f^{-1} o\ f = e$(the identity function)for all Elementary functions.Other facilities,some of which are optional,include:expansion of multiple products of sums,or of exponentiated sums,using (non)commutative multiplication,with possible(arbitrary,rational)bounds on exponents of terms retained;conversion of rational expressions to CRE form,with various,optional'splittings';reduction of certain expressions containing logarithms,exponentials and radicals(all functionally equivalent forms of subexpressions being replaced by one common form,so that the difference between equivalent functions is always simplified to zero:see Section 13 for more comments on this problem);and,partial fraction decompositions for the rational parts of expressions.These are just a few of the vast array of simplification operators available;full details may be found in the manual(though the underlying mathematical algorithms are not discussed,and may be changed whenever a new,more efficient procedure is found).

The selection functions, allow identification and/or replacement of any (sub)-expression,the main operator always baring the label 0,and the k th factor in any product,the label k;while, in quotients,the numerator and denominator have labels 1,2,respectively,etc..Any expression can be 'atomized' in this way,though the precies details reuqired to avoid ambiguity must be formulated with care.Examples of such procedures are the tabulation of parts of an argument expression, and the insertion or removal of parentheses.MACSYMA is particularly elaborately endowed with selection and dissection functions(details in the manual),giving users wide possibilities for close scrutinyof stored expressions.

Further internal procedures for simplification are aimed at the minimization of all LISP expressions.For instance,the sum of A,B,C,...K(in any order)has the representation (PLUS A B C ... K);square roots are put in the form $E^{1/2}$ (for all suitable expressions,E);differences are represented as X +(-1) * Y,and quotients as X *Y^{-1}.Expressions are ordered via their subexpressions being ordered first—a recursive procedure—all variables in the subexpressions being ordered alphabetically;and one puts (numbers) before (constants),and (constants) before (variables).In this scheme,functions have the ordering of their arguments(comparing first arguments first,second arguments second,etc.)If this fails to distinguish them,then they are compared via their names.All of this may appear somewhat pedantic;but it is,in fact,essential if uniqueness of representation is to be achieved—and,without such uniqueness,effective simplification is virtually impossible.Analogous strategies are used in all systems.

Many other facilities are concerned with controlling or processing the current stored expressions during a program;with the printing out of storage management information,or of the net time used.The 'SAVE' facilities allow data to be preserved and manipulated in files(so that,for instance,expressions can be simplified before being used in the next stage of a calculation.Some general remarks on simplification(especially in relation to.the mathematical problems raised) are made in Section 13.

2.5 Basic mathematical functions.

The user may apply arbitrarily complicated operations to general functions,using the processes of algebra and analysis;those results the system can simplify are simplified.A wide selection of examples is given in the manual.Underlying the general operations are various basic functions,many of which are found,in some form,in all systems.For MACSYMA,the basic functions are as follows(where expr,arg,mean,respectively,expression,and argument).

ABS(X):= $|X|$;FLOAT(expr)(converts all numbers in expr to floating point form); ENTIER(X):= $[X]$;SIGNUM(X):= $X/|X|$,for nonzero,real X;and := 0,if X is 0,complex (not real),or non-numerical);MIN(args)(gives the minimum of its--real--arguments); MAX(args)(gives the maximum of its--real--arguments);SQRT(X)(= $X^{1/2}$); EXP(X):= e^X ;LOG(X):= log X ;LOGEXPAND(X)(replaces all logarithms of products by sums of logarithms,for terms in X);LOGSIMP(X)(simplifies,if possible,all factors in X of the form exp $\{f[\log X]\}$);DEMOIVRE(expr)(puts e^{A+iB} into the form P + iQ,if such an exponential occurs in expr);PLOG(X),GLOG(X)(these produce, respectively,the principal branch of the multivalued function,'log X',and the multivalued function itself(as a collection of branches));BINOMIAL(X,Y):= X*(X-1)... *(X-Y +1)/ (Y*(Y-1)*...*1);GENFACT(X,Y,Z):= Π (X-kZ) : $0 \leq k \leq Y-1$; RANDOM(X)(a'random integer'between 0 and X-1);FIB(X)(the Xth Fibonacci number,if X is a positive integer—otherwise,undefined);GAMMA(X)(the Gamma function for argument X);BETA(X,Y)(the Beta function for arguments X,Y);ERF(X)(defined by dERF(X)/dX $:=(2/\sqrt{\pi})e^{-X^2}$,ERF(0) = 0);EULER(X)(the Xth Euler number);BERN(X)(the Xth Bernoulli number);ZETA(X)(gives the value of Riemann's Zeta function,for certain integer arguments);PSI(X)(the logarithmic derivative of the Gamma function at X);and, all of the trigonometrical and hyperbolic functions,and their inverses.

Even this list is not quite complete,and,in any case,new basic functions may be added to the system at any time.However,the purpose ofgiving such a list is to illustrate how wide the range of the system is at this basic level.By adding facilities for operations in algebra and analysis,backed by an array of 'system functions',the designers have built a system which,with skilled use,can enlarge the scope of mathematical investigations considerably(as I argue in Sections 14 and 15 of this paper).In the following paragraphs,outlines are given of the principal mathematical facilities in MACSYMA,together with a bare indication of the range of system functions(aids to prograaming and the manipulation of data).

2.6 Sums and products.

These functions cover all of the standard manipulations for finite sums and products,taken over finite sets of elements,arbitrarily indexed.Special features include Cauchy mutiplication(for the product of two infinite series—when it exists), certain closed-from summations obtained by 'splitting methods'(Gosper(1976,1979)), and the 'first backward difference operator'.

2.7 Differentiation and integration.

All of the standard manipulations for differentiation are covered(incuding the use of 'formal derivatives',where the operand is given only formally,so that no evaluation is involved--only the production of 'df/dx' from f).Leibniz' rule for differentiation of products can be invoked.Special facilities for differentiation of tensors,and for exterior differential calculus are available; these will be discuused briefly later.

The most remarkable procedures are RISCH(invoking the Risch decision procedure for indefinite integration--see Section 12);the LIMIT procedure(with syntax LIMIT(expr,var,val dir),to find the limit of expr as the variable approaches the value val from direction dir.See Wang(1971),where this routine is used to study the evaluation of definite integrals in MACSYMA--especially,contour integrals, using the residue method.Notice that,some definite integrals give well-defined, closed form resultseven when the corresponding indefinite integrals do not exist at all).An alternative limit program to Wang's is based on the use of Taylor series.Another routine,ODE2,with syntax ODE2(diffeq,depvar,indvar),solves ordinarydifferential equations of the first or second order,returning either an implicit or an explicit solution--or else,returning FALSE if no solution can be found by the routine.A battery of methods provided by a wide variety of users of MACSYMA makes up this package.Some types ofequations of higher order can be handled,too; and no doubt this facility will be extended as new methods become available.

2.8.Substitution functions.

There are several of these functions,allowing a variety of basic replacements(either in conjunction with 'PART',which identifies the part to be replaced;or else within a given class of expressions—e.g.,rational expressions.The replacement of radicals by suitable rationalizing substitutions is also possible.The facility 'ATVALUE' is used to assign boundary values at given points;and'AT' will evaluate an expression at specified arguments.These substitution procedures,used in conjunction with other mathematical operations,allow the user to implement intricate procedures;but care is necessary,since substitution does not always commute with other operations with which it is combined.

2.9 Functions for rational expressions.

This covers all expressions that are representable as quotients of two polnomials,in any number of variables.Distinguished variables are identified,and the functions are treated as rational functions of a single variable,whose 'coefficients'are polynomials in the remaining variables.A listing function,'RATVARS', list all of the variables,from main to least main,for future calculations.The basic function 'RAT'converts its argument toCRE form(including the replacement of all floating point numbers by rationals,to within a prescribed tolerance,and elimination of common factors of numerator and denominator).Several simplification routines are designed to standardize rational expressions.Other functions return the numerator or denominator;assign weights to each variable(so that,e.g.,all expressions of the same total weight may be listed,or the term(s)of maximal or minimal weight may be extracted);compute quotient and remainder of two polynomials,relative to the main variable;compute the greatest common divisor of two polynomials; determine the resultant of two polynomials relative to the main variable(though settings for other variables are possible,for all of these calculations).There are also routines for differentiation of rational expressions—the pointbeing that a special method is used,which is much faster than the general 'DIFF'routine.(For integration,the 'rational case'is not treated separately from the general routine:see Section 12 for comments).
Closely related to the functions just mentioned are others for dealing with algebraic integers(solutions of monic polynomial equations(over a given field)with integer coefficients),and with extended rational expressions(truncated power series with rational functions for coefficients).Very few explicit routines are designed for these objects,but a judicious use of the 'WEIGHT' and 'EXPAND' facilities,together with 'TAYLOR'(which produces a Taylor series form of a truncated powerseries about preassigned base point(s),with a symbolic remainder term), covers most contingencies.For calculations withgeneral algebraic numbers, the leading system is probably SAC-1,which has a comprehensive set of routines.However,MACSYMA also covers this area quite well.

2.10.Solution of algebraic and transcendental equations.

There are functions for solving implicit or explicit equations,or sets of equations.Examples include:determination of the number of real roots of a univariate polynomial in a given interval;of all of the roots of a real univariate polynomial;of the solutions of sets of linear equations,or of polynomial equations —which may be nonlinear—and of the solutions for general cubic or quartic equations.Several other facilities related to factorization,and some provision for dealing with special types of transcendental equations,involving logarithms and exponentials,are also included.Most of these routines are due to Stoutemyer(1975).

2.11.Matrix functions.

Anextensive range of functions for matrix calculations is built into MACSYMA.These functions include:evaluation of the characteristic polynomial of a square matrix; listing all the minors of a matrix;producing the 'echelon form'of the argument matrix;computingthe determinant of a square matrix,in symbolic form(there are two algorithms for this;as well as special algorithms for sparse matrices);and,finding the rank of a matrix.There are also facilities for generating a matrix from elements stored in the system,for doing standard matrix algebra(e.g.,finding inverses),and for handling 'Kronecker products'.

2.12.Functions for 'Poisson series'.

The typical term of a Poisson series(actually,only finite sums are considered)is of the form (general MACSYMA expression) x (sine,or cosine),where the argument of the trigonometric functionmay be a linear combination of up to six prescribed MACSYMA variables—though this last restriction could be removed.For some purposes, the periodic and nonperiodic parts of the series are treated separately.For instance,differentiation and integration are limited(as special facilities)to only those variable not appearing in both parts of a term of the series.Special substitutions are provided,with full reduction of all trigonometrical functions to linear forms.The two types of factor in any term can allow substitutions only in the variables assigned to them for differentiation,etc..General expressions which include trigonometrical functions may be rearranged as Poisson series,and all of the dissection or expansion facilities may be used in this setting.Various trigonometric simplificationroutines are available for optional use.

2.13.Functions for Taylor series.

The manipulation of Taylor series is one of the most important facilities in any symbolic manipulation system,subsuming many different kinds of operations.For instance:determination of local power series expansions for (multi-variable) functions(about speciefied base points);direct analytic continuation along a curve in the complex plane(which can be put in an effective form for approximation--see Henrici(1974));substitution of one power series into another;and,determination of he Taylor series for a (convergent)infinite product.(Of course,only a finite number of terms of any series can be obtained,but this number can be chosen arbitrarily--up to a high bound,depending on the machine used.Moreover,in certain cases,a general formula can be found for 'the Nth term').

For multivariate expansions,a technique is used which caters for possible interdependence of the expansion variables,and for the occurrence of singularities, limiting the range of validity of the expansion.Series in negative powers(useful when asymptotic properties are studied) may be obtained directly from the argument function.This includes the possibility of finding expansions 'about the point at infinity'.Examples of the use of all these facilities are given in the manual.

2.14.Laplace transforms.

In MACSYMA,Laplace transforms may be found for functions involving(at worst)the basic components:EXP,LOG,SIN,COS,SINH,COSH and ERF.Eventually,radicals,elliptic functions and other,more exotic function will be allowed(when the general integrations procedures have been extended adequately--see Section 12).As it is,there is still a wide range of problems for which the Lapce transform facility is useful--including the solution of (systems of)ordinary differential equations with constant coefficients,and of 'convolution integral equations'.Rational functions may be covered by the basic routine,and,by using the representations of Special functions interms of generalized hypergeometric functions,Laplace transforms of 'original' having Bessel functions,etc.,as factors,may be evaluated.See Avgoustis(1977)for details.Presumably,facilities for handling Fourier transforms can be based on these algorithms,but this has not been done explicitly in MACSYMA.It appears to be the intention to cover all of the results given in the 'Bateman Manuscript Project'collection of Laplace transforms(Erdelyi(1954)),as well as others not given there.All of this amounts to a very powerful facility,with wide application in pure and applied mathematics.

2.15.Combinatorial functions.

These comprise mainly transformations of binomial,Beta and Gamma functions to (generalized)factorial forms—or,the reverse procedures.Another routine generates the Bernoulli polynomials up to any prescribed order.

2.16.Continued fractions.Number-theoretic functions.

These functions include:generation of sequences of convergents for the continued fraction representation of a numerical argument;conversion of a list form of continued fraction into the standard form.There does not appear to be any explicit facilityfor treating analytic continued fractions(such as may arise,for instance,from Stieltjes integrals involving a parameter),but it should not be difficult to handle such problems,using other general routines in MACSYMA.The number-theoretic functions return the Nth prime(with upper bound 489318);find Euler's Phi function(for positive integer argument);compute the Jacobi symbol of integers P,Q (of importance in the theory of quadratic residues);and,solve Pell's equation ,in the form $A^2 - nB^2 = 1$.The Nth-power divisor function is given,also.

2.17. Tensor manipulation.

There are two levels of tensor manipulation in MACSYMA:explicit,and indicial.In explicit manipulations,the tensors are stored as arrays,and all operations on the tensors correspond to explicit(algebraic/analytical)operations on the elements in these arrays.In other words,the actual result of(say)covariant differentiation or contraction on any given tensor(s) may be recovered,and displayed as a functional array,derived fromthe original one.On the other hand,the results of indicial manipulation merely keep track of operations performed on given tensors by manipulating(possibly,altering)their indices,and expressing them symbolically in terms of other indexed entities.However,no functional forms are ascribed to any of these symbols during the manipulations;and,at the end,the form of the resulting tensor is also unknown.

In calculations,both types of manipulation are useful.Initially,the indicial operations may be used to effect simplifications and reductions of expressions that otherwise would be almost too unwieldy to fit into the system at all.After this, explicit operations may be able to produce results that are of immediate significance in some problem.The main area of application is Riemannian geometry,the metric tensor being prescribed at the outset.Many calculations in relativity(and some in elasticity)are of this type.Indeed,all of the operations required in typical relativistic calculations(determination of the Christoffel symbols,the Ricci and Riemann tensors—and hence,of the field equations,for a given metric) are provided,as well as various manipulatory functions(for raising and lowering indices,contraction,reducton to canonical form,etc.).See d'Inverno(1978)for details of symbolic computingin relativity,and Section 11,for comments on special-purpose systems for relativity.All of the MACSYMA operations are described in the manual, with indications as to which combinations of them can be used to particular advantage.

2.18.Miscellaneous system functions.

Most of the remaining functions are used for programming,or handling data,and the prospective user must,of course,become familiar with these facilities.However,the attention here is confined to giving an idea of the scope of MACSYMA,by listing the function types,with brief indications of their principal uses,when this is not obvious.

There are functions for: declaring and assuming;for establishing contexts(in information retrieval);for list-handling(including several LISP-like functions); for specifying properties(e.g.,to set-up,display or remove a property);for handling internal properties(e.g.,'DECLARE',for specifying the types of variables in use);for handling user properties(e.g.,'PUT',which allows any atom to be given an arbitrary property);for type-testing(e.g.,identification of'atoms');for general pattern-matching(to test expressions for combinations of(specified)syntactic and semantic patterns--e.g.,to aid in simplification;but also,more generally);for utility,I/O and display(including:de-bugging,options,output format,storage clearance,file-referencing,ordering,translation and compilation);for editing(e.g.,removal of syntax errors,interactively);for batch processing;for storage and retrieval(mainly,file-handling);for plotting(e.g.,scale plots of contour surfaces,in two or three dimensions;and,for error detection(including tracing functions--for localizing errors;identification of predicates that were not evaluated,and checking of assignments for variables,arrays and other entities in a program).

There are are,of course,several functions of each type,and full details and examples are given in the manual.However,this bare list does illustrate the wide variety of types of functions which(in conjunction with the mathematically oriented facilities)must be deployed--with some semblance of optimality--in the construction of an efficient,general-purpose symbolic computing system.The task of producing adequately expressions (using,e.g.,the methods mentioned in Section 2.3), increases the complexity of the design problem even more.It is not surprising, therefore,that the various systems differquite markedly in their dominant characteristics--which makes direct comparisons among them somewhat inappropriate.Nevertheless,it is important to have some criteria for meaningful comparison,since potential users do not have the time or opportunity to try their calculations on each system,in turn!For this reason,some tentative remarks on comparison are made in Section 16.Bibliographical details relevant to MACSYMA may be found in Section 17,and in the list of references.Samples of I/O,and of programs,are given in Section 13.

3. ALTRAN

3.1. Preview

This system was developed by W.S.Brown and others at Bell Telephone Laboratories—initially,between 1961 and 1966;the current,portable version was designed in the period 1968-1973.Its basic capability is to perform rational operations on rational,n-variable expressions with integer coefficients.These restrictions are not as drastic as they may appear at first glance,since(as it is argued in the manual), a fullalgorithmic treatment of such expressions is possible(see,e.g.,Knuth(1968,1971)), and many practical problems may be formulated using only rational functions(and, in any case,many classes of nonrational functions have sufficiently good rational approximations—even,arbitrarily good ones,in several important cases).Moreover, often,both simplification and evaluation problems for nonrational functions may be studied most fruitfully by regarding wider classes of functions as algebraic or transcendental extensions of fields of rational functions(indeed,the Liouville-Ritt-Risch algorithm for indefinite integration(see Risch(1969))proceeds in this way—as do many analyses of algorithmic complexity for processes involving non-rational functions).Thus,the potential scope of ALTRAN is broader than one might imagine,given only its basic specification.

The system can be implemented on any sufficiently large computer with a standard FORTRAN compiler.It requires about 240 K-b(270,with 'workspace')on an IBM machine, and about 36 K-words(44,with workspace)on a Honeywell.The syntax is that of FORTRAN,with some extensions.Like FORTRAN,it is a one-case language.The collection of FORTRAN routines on which the original version of ALTRAN was based,was known as ALPAK,and ALTRAN allowed these routines to be called in a systematic way.However, this terminology is no longer used:the current version of ALTRAN is regarded as a self contained package,whose 'language' is an extension of FORTRAN.Direct communication between ALTRAN and FORTRAN has been implemented partially.This is useful in numerical phases of calculations.ALTRAN is highly portable,but it exists only in noninteractive form(though interactive program development,and submission of 'batch jobs' is possible).

ALTRAN(like FORTRAN,ALGOL and PL/1)is a procedural language(i.e.,a vehicle for expressing procedures,algorithms,etc.).Many of the 'keywords'(reserved expressions)are used to denote procedures and,as such,appear frequently in the listing of ALTRAN programs.There are also various system functions,and a library of mathematical routines,whose underlying algorithms may be standard,but are,in several cases,original(and reduce the running time of programs).For instance,the effective determination of greatest common divisors,for pairs of polynomials in several indeterminates,is an important step in many algorithms used in symbolic computation;and some of the pioneering work in this area(Brown(1971))is incorporated in ALTRAN.(Collins(1967)also made important contributions:see Section 8,on 'SAC-1').

3.2 Overview.

The following basic features give an idea of the mode of operation of ALTRAN.

→There is no'interaction'between the numerical(floating point)and symbolic(e.g., literal)data,except in the(final)evaluation of definite expressions.

→ Rational expressions(and numbers)may be reduced to'lowest terms'.

→ Zero cannot appear 'in disguise'(i.c.,as a(complicated but)identically vanishing expression).

→ All expressions generated are automatically well defined(i.e.,the syntax rules are applied at all times).

→ The syntax of ALTRAN is essentially that of FORTRAN;but,semantically,ALTRAN has much in common with PL/1.

→ No computation can yield an incorrect result,though it may be stopped for lack of space or time.(This should be interpreted as meaning that any error will stop the program—either at compile time,orelse,at run time;so that,if a final result is produced at all,then this result cannot be wrong.However,machine error is not ruled out,even if it is improbable;so,as in all computing,caution and common sense remain indispensable!).

→ All syntactic and semantic rules are always enforced(as mentioned already,for syntax).

→ All syntax errors are detected at compile time,if possible(otherwise,at run time).

→ A program can regain control after all but the most serious errors(specified in the manual)have been detected(and corrected).

→ 'Back-reflection of errors'(from a given procedure to the one invoking it)is automatic.Thus,even if an error is missed initially,it will be detected eventually.

→ Serious errors are consigned to a 'problem-oriented dump',for subsequent analysis,if a program is stopped altogether.

These modes of operation,some of which are overlapping,are treated fully in the manual,where the system is described on several levels of completeness,including the full range of types of statements,with examples of their uses.Most of these types of statement are familiar from FORTRAN,with minimal modifications to acconmodate the extra symbolic facilities.Correseponding classes of statements exist in all symbolic computation systems(though those based on LISP have some types of statement for which the FORTAN-based systems have no counterparts).A partial interface with FORTRAN is provided,and inter-procedure communication within ALTRAN is also possible.

3.3 The library routines.

A complete list of library procedures is given in the manual,with some examples. Here,it suffices to indicate the scope of each type of procedure,and to outline a few,typical facilities.There are procedures for: I/O;numerical computation; test and conversion;algebraic analysis;algebraic computation;options,time and statistics;error-handling;array operations;modular reduction;matrix computations; and,truncated power series computations.Brief observations on each of these types of procedure will be made now.

3.3.1 Input/Output.

There are two integer types,'short'(one machine word)and 'long'(two-to-ten words). Similar conventions are used for rational numbers,and for the coefficients appearing in algebraic expressions.Formally,'integer','rational','real','algebraic' and 'logical' data types are distinguished(essentially,according to the types of numbers or variables of which they are composed).All variable types must be declared. Compound objects,such as lists,acquire the type of their component parts.(Labels may also be regarded as a separate variable type,when this is relevant).The 'read' and 'write' instructions govern the actual input and output of data.Further facilities allow various sorts of output to be obtained.

3.3.2 Numerical computation.

All input(and output)here consists of integers(unless it is stated otherwise).The main facilities include:determination of the greatest common divisor,maximum and minimum(of two integers);of the numerator or denominator of a rational number; of the quotient and remainder in the division of one integer by another(which includes the 'congruence definitions',via $c \equiv d(\mathrm{mod}\ r)$);of the 'modular reciprocal', d,as the solution of $cd \equiv 1(\mathrm{mod}\ m)$;of a prime larger(or smaller)than a prescribed integer,n—in a certain range;and,of the values of most of the Elementary functions and their inverses(by invoking the corresponding FORTRAN facilities).

3.3.4 Test and conversion.
These facilities either test(an integer)for some property--and give the value 'true' or 'false'--or else,they convert the(integer)input into some specified type(e.g.,long,short)The forms of input may include rational or real(floating point) numbers,each being put into some special form;but the basic facility is for integers.

3.3.5 Algebraic analysis.
Among the main procedures here are those for:defining the 'layout' of an expression(e.g.,the size of n-dimensional arrays);for finding the number of(distinct) factors in an expression;the sum of the numbers of nonzero terms in all(multi-variable polynomial)factors in the input expression;and,for finding the degree(in a specified variable)of the input expression.Other facilities include:return of the numerator(repsectively,denominator)of a rational expression;full expansion of a product of polynomial factors,in several variables;identification(and subsequent removal)of factors of the form $f_i^{c}i$ from a given expression;return of the term(s) of lexicographically largest(respectively,smallest)exponent in an expression;return of the term of degree 0(or else,of 0);and,return of x^{-1}x(sum of terms of degees between 1 and h—a prescribed integer).One other basic facility involves the concept of'levels of canonical form'(pioneered in ALTRAN),which is defined as follows.A rational expression is canonical if its numerator and denominator are relatively prime.A polynomial is normal if its leading coefficient is positive,and its coefficients form a relatively prime set.The levels are:(1)normal and canonical,(2)normal and 'alleged canonical',(3)normal,but whether canonical unknown,and (4)special(=single polynomial factor,not necessarily normal).Although these classifications are somewhat technical,they do illustrate the care with which ALTRAN has been designed.In these terms,the other basic facility is:identification of the level of canonical form of the input.(The main use of this classification is in specifying corresponding forms for the results of calculations;again,certain sorts of computation may be easier for arguments in particular forms;and so on).

One important matter(which must be raised in relation to every system)is its 'knowledge' of the Elementary functions.For numerical work,ALTRAN shares the FORTRAN facilities.However,on a symbolic level,the system knows nothing about the Elementary functions(unlike many of the other systems),and special subroutines must be used to 'teach the system'to apply the relevant basic rules--e.g.,for differentiation and integration.

3.3.6 Algebraic computation.

The most useful facilities here are:transformation of any input expression to canonical form(level k);return of the greatest common divisor of any pair of input expressions(i.e.,not just polynomials);repeated or mixed partial differentiation(which may be regarded as essentially 'algebraic'when restricted to rational expressions);multiple integration of(in effect)polynomials in n variables;and,return of prescribed 'blocks'of terms from an input expression(for separate analysis).Notice that the integration procedure is minimal,and does not even cover general rational functions of one variable,though it would seem that such a facility could be provided.
The other basic aspects of general algebraic computation are concerned with simplification and substitution.Apart from the automatic use of obvious cancellation and elimination procedures,the imposition of side relations(such as '$i^2+1 = 0$,for complex numbers;or,at the other extreme,arbitrarily complicated identities that happen to be relevant to particular calculations)is possible.In effect side relations entail substitution of one expression into another--one of the most crucial facilities in any symbolic computation system,subsuming as it does virtually all general transformations in analysis--so that many nontrivial programs make essential use of it.ALTRAN is less well equipped with dissection functions than are some of the other systems,so there is probably a fairly low limit on the complexity of expressions for which it can handle substitution and simplification effectively.Moreover, such procedures create enormous demands on storage space,so they must be used with caution.

3.3.7 Options,time and statistics.

The emphasis here is on the setting of 'preferred actions'in various contexts.For instance,it may not be desirable to reduce all rational expressions to lowest terms(the 'default setting');or,one might wish to expand allpolynomials fully in one part of a calculation,but to retain them in shorter forms in another part.
Other examples of these options(or 'switches')include the choice of levels of canonical forms,'modular bias'(for congruences),and so on.Naturally,every system has options(they are especially profuse in MACSYMA),and they should be used rather like the stops on an organ,if the system is to be exploited to full advantage.Several kinds of output option are available(though,ALTRAN has relatively few,and only one-dimensional output is produced).Among the other facilities are:return of the time used(over any selected part of the program,or for the whole program), which is useful in program optimization;return of various 'statistics',such as the maximum number of words used(taken over several runs);current number of words used;and,contents of the 'problem dump'(which contains those parts of a program apparently causing it to stop--and corresponding output generated,etc.).These facilities are of use primarily to the system designer,but they could also be used by programmers,trying to correct and then 'tune' a complicated program.

3.3.8 Error handling facilities.

This includes such things as:returning the error number of a current procedure(if it has an error);verifying the consistency of use of 'workspace'(the active area of storage space where current computations are performed);. various syntax checks;and,printing out the contents of the workspace at any time(in a form relevant to system designers).Details are given in the manual—and there is some variation according to the implementation concerned,since each type of computer has its own built-in facilities for error detection(a remark that applies to all systems considered in this paper).However,for ALTRAN,this kind of variation is practically negligible.Again,in general,facilities for interactive systems are more extensive than for the others(e.g.,in the crucial matter of syntax checks).ALTRAN does have some checks at the input stage(as explained in Section 3.2),but interactive systems make the correction of errors and the improvement of program strucure as simple as possible.

3.3.9 Arrays.

These facilities,of special relevance for matrix and tensor manipulations(but not confined to this area)may be summarised as follows.

→ Given any n-dimensional array,where the 'j th dimension'has range r_j,with upper and lower points u_j, ℓ_j,return n,u_j, ℓ_j,r_j,and $\prod_j r_j$.

→ Return the total number of elements in an array.

→ Set-up in the store any specified,general array(the range of each subscript—or set of subscripts—being given).

→ Transform each element of a given array to. algebraic canonical form(level k).

→ Return an array formed from the descriptor block(dimension specification)of one array and the value block(specification of the values of elements)of another array.

These are formal operations,having no direct significance for linear algebra;but their application to the generation and manipulation of matrices and tensors is obvious.

3.3.9 Modular reduction.

These facilities extend the ones already described for integer variables.

→ Reduction of a given rational expression, α,so that all x-dependent factors are reduced modulo β(essentially,a polynomial in x).All arguments(or their factors) are viewed as polynomials in a distinguished variable,their coefficients being (rational)functions in the remaining variables.

→ The quotient,remainder and resultant of a given pair of polynomials,say, γ, δ,in a distinguished variable)are returned;also,the reciprocal of γ modulo δ.

→ There are operations on the coefficients of polynomials occurring in rational expressions,the results being reduced modulo a specified integer,m.

3.3.10 Matrix calculations.

A very brief indication may be given of these facilities(though the crucial point is the quality of the underlying algorithms—and this is,mostly,high in ALTRAN). The routines include:transposition;matrix product;inner and outer products of vectors;determinant and inverse of square matrices;and,solution of systems of linear equations.Combinations of these facilities to implement more complicated algorithms are,of course,also possible.There do not appear to be any routines specially designed for tensor manipulations,but these applications have been implemented for many computations in relativity and theoretical physics.

3.3.11.Truncated power series(TPS).

The TPS are,typically,obtained from terminating Taylor series for(multi-variable) rational expressions(though there are many other contexts in which they can arise naturally—see,e.g.,several of the applications in Section 15,where the emphasis is on general approximation procedures).Although a TPS is formally a polynomial,the two concepts are quite distinct,since rational function division is not (formally) identical with power series division,and the TPS represents merely one stage in a hierarchy of possible approximations.(See,e.g.,Knuth(1969),Henrici(1974)). ALTRAN has a particularly wide range of TPS facilities,including the following. Returnofsumand difference of two TPS;returnofTPSrepresentation (to given order(s)) of any rational expression;division of one TPS by another;raising of a TPS to a given(positive integer)power;differentiation or integration of a TPS(all coefficients being treated as constants);substitution of a TPS into a rational expression—or,into another TPS;reversion of a TPS;evaluation of a TPS at a symbolic value;and,return of the order(in a distinguished variable)of a TPS.

To illustrate how effectively these facilities may combined,consider the functional equation $g(\xi) = f(x)$,where ξ is a TPS in x.The formal ALTRAN solution is: 'x = tpssbs(tpsrev(f),g)',which gives x in the form $x = f^{-1}(g(\xi))$. In other words,the TPS for g is substituted into the reversion of the TPS for f.

The documentation for ALTRAN(covering installation,maintainance and programming) is excellent.The manual is well produced,and deals with all facets of operation of the system;it includes a bibliography.

4. ANALITIK.

The ANALITIK/MIR sequence of packages is being developed under the general direction of V.M.Glushkov,at the Institute of Cybernetics of the Academy of Sciences of the Ukrainian SSR.Computer scientists at the Helsinki University of Technology also play an active part in developing facilities for ANALITIK.
The system,which is interactive,appeared in its initial version in 1965,with a permanent form in use by 1968.The I/O language has some similarities to ALGOL,but many extra features.The crucial difference between ANALITIK and all other systems is that its main facilities are microprogrammed--in other words,they are realized as part of the harware design.Thus,they cannot be varied—only applied(as a tool) to mathematical problems.This characteristic of ANALITIK persists in all implementations on MIR computers,each computer corresponding to(indeed,embodying)the latest version of the language.So far,there have been three phases of development,and a fourth is in prospect.These phases may be described,briefly,as follows.
Phase 1.MIR-1(1968)is a very small(and slow)machine whose main advantage lies in the natural form of I/O language(for instance,summation may be performed directly, by writing Σ ...,without the use of 'loops')but the analytical scope is small,and use is limited to fairly simple problems.However,the basic features of the system are retained in the later models.

Phase 2. MIR-2(1969) extends MIR-1 to cover more general analytical transformations and more effective interactive use;but it is still slow(about 12,000 operations per second--since arithmetic is done in decimal,rather than binary form),and the memory is extremely small(only 8 K-words).

Phase 3.MIR-3,ANALITIK-74(1974-?), introduces several essential improvements, including:larger memory(64Kb— still not comparable with those in the other systems),simpler semantics,more compact notation,better simplification and editing facilities,higher speed(from 10 to 50 times as fast as MIR-2,depending on the type of calculations),and more 'primitive instructions'(the basic tools of the package, of which there are,essentially,only five in MIR-2;and these are too general for typical use inapplications).On the operating level,the main improvements are:possibility of 'stepwise running'of test programs(with automatic intermediate output); interactive treatment of syntactic,semantic and other errors(so that the calculation may be resumed at the start of the corrected portion);and,determination of the status of the machine,at any time(giving,e.g.,the current values of all variables,and stating whatoperation is currently being performed).

Phase 4,MIR-4,is not completed.It can be expected to include more advanced micro-programmedelements,higher speed of operation,more flexible interaction,virtual memory,history file,etc..In principle,it is clear that these improvements can be made;but it is well known that major advances in this field tend to take years, rather than months,so it may be a long time before this enhanced ANALITIK system becomes available.

For the present outline,it is the unique construction of the MIR-n range of systems that is stressed(with the reflected versions of the language,ANALITIK,of which the computers are partialrealizations).The basic symbols used are Latin and Russian capitals(though an 'English'version,'E-ANALITIK,has been introduced at the Helsinki University of Technology;and this will,presumably,be extended to cover the latest forms of ANALITIK,as they appear).

4.2.Structure of the system.

There are just five basic operations governing analytical manipulations—but these operations are very general,and may be used to construct complicated transformations.This is in line with the design philosophy of providing fixed tools for efficient manipulation,as opposed to a collection of ready-made routines.It is these basic tools that are implemented in hardware.The basic operations will be listed now.

→ Simplification (into any of three 'canonical forms').

→ Comparison of expressions (an elaborate 'pattern recognition' facility).

→ Application (of transformations,subject to user-defined rules:a key facility).

→ Formal differentiation.

→ Formal integration.

For any calculation,the main program has the structure:

LET $\lambda . Y_1 ; \ldots Y_n$ WHERE $M_1 ; \ldots M_k$

where λ is a label,the Y_i are statements(or operators)and the M_j are descriptions (or definitions).Extra descriptions(e.g.,initial data or boundary data for differential equations)may be introduced by means of the construction:

LET $M_p ; \ldots M_q$ END. Operations are activated by a DO or EXECUTE command. There is a wide range of mathematical symbols,together with several combinations of special parentheses(used for separating expressions into'components'.Internally, expressions have a 'tree representation',corresponding to the Polish logical notation(see,e.g.,Lukasiewicz(1963)).

4.3 Remarks on I/O and some basic procedures.

The ANALITIK language has many attractive features because of its mathematically oriented form.However,only one-dimensional input and output are available(though, possibly,an option for two-dimensional output will be included in the next version).In spite of this limitation,the formation of expressions is quite simple,for instance,the expression

$$\prod_{n=1}^{2p} \sum_{\ell=1}^{n} \int \{\alpha x^{\ell} + \frac{\partial}{\partial x}[\sqrt{\ell}x^{m}\cos(x/\beta)]\}\, d x \quad \text{becomes}$$

```
Π(N=1,2×p ( Σ ( L= 0,N ( ∫ ( A × X↑ℓ + ∂/∂X(√(L)×X↑m × COS(X×√(A))))dX )))).
```

Any of three arithmetic modes(integer,rational or decimal)may be specified.Variables may have as their values any well-formed expressions;or else,they may be defined by their descriptions.Otherwise,they just stand for themselves(atoms). One special variable has no formal description,but its(fluctuating)value is the current content of the working zone (where current computations are performed).In ANALITIK,the contents of the working zone may be viewed on a screen,and altered,not only by normal editing procedures,but also by means of a 'light-pen',for underlining—a form of extra identification.(As mentioned in Section 0,some of the early,experimental systems also had this device).The working zone is the operand for all analytical transformations .Arrays may be specified formally(e.g.,as A [11] ,for a vector),or explicitly(by listingall of their elements).There are standard functions(e.g.,Elementary functions)and user-defined functions(of the form F(X1,...,XN) = (expression)).These functions may be specified at the outset, but they could be introduced during the computation.

Substitution in a function already defined is achieved by altering its arguments directly.For instance,if one has specified a function,f,of variables x,y,explicitly,then the substitution x = g(u,v),y = h(u,v)—with g and h also explicitly given—is accomplished simply by writing F(G(U,V),H(U,V)).This notation is very compact,and compares favourably with that in most other systems.Operational notation for differentiation and integration(and most other analytical routines)is simple:for instance,one writes ∫(E),or ∂/∂x ↑k (E) ;also, √(E) , SIN (E),etc—as in the above example.On the other hand,factorials are written in full(as products of all integers from 1 to n),and binomial coefficients as quotients of factorials—in line with the use of the system as a tool-kit,from which facilities should be constructed.Nevertheless,the system does treat the Elementary functions directly (unlike ALTRAN—or SAC-1;see Section 8—where special,miniature subroutines are required).Conditionals have the form IF ... THEN ... OTHERWISE ,and the logical connectives AND , OR ,are used,together with NOT for negation.

Simplification procedures are similar to those in many other systems(though not so extensive--unless extra routines are put in by the programmer).There are three 'canonical forms' of simplification,the second('intermediate')one being the default setting.However,all of these forms are very basic,and many additional conditions must be imposed(as small subprograms)for effective reduction of complicated expressions in a particular context--clearly,the use of side relations along with substitution plays an important rôle here.Details,and examples,of all operations for ANALITIK may be found in the references given at the end of this Section.

4.4 Operations on equations.

It is in the rôle of equations that ANALITIK differs most strongly(on the software level)from the other systems.This is because,for ANALITIK,equations are the basic entities from which generaltransformations are constructed.By allowing a variety of operations to be applied to equations,the designers compensate to some extent for the lack of built-in routines.In view of this variety of operations on equations,the user must declare all transformations and their applications, in detail.One of the most basic operations,in this context,is the 'USE' operator, which effects substitutions,by replacing the left member of an equation(say, $E_1 = E_2$)by its right member--within another(specified)expression,E,containing E_1 as a subexpression.

Three types of variables are defined:simple(which cannot be replaced by others without invalidating the equation;parameter(which may be replaced by any expression in a broad class--as in identities);and,separate,or 'functional',(which may be replaced by any other variable,but not by the values of variables).Typically, separate variables occur in differentiation and integration operations.The question of 'value' here,involves a distinction between(e.g.)descriptions of expressions and the possible values they could assume after various substitutions are made.On the whole,such distinctions do not affect users directly,but failure to observe them may lead to program faults.

There are several ways of naming equations,ranging from the pure identifier(label) to the explicit representation($E_1 = E_2$).Systems of equations are handled analogously.The very small memory of MIR-2(which,as mentioned already,is enlarged somewhat in MIR-3)makes it essential for(collections of)complicated expressions to be manipulated with minimal use of storage space.For this purpose,the 'value of an equation,$E_1 = E_2$' is a useful concept.It is defined to be the resulting equation (say,$F_1 = F_2$) obtained when all evaluable parts of E_1 and E_2 are evaluated as fully as possible.The equation,$F_1 = F_2$,then forms a basis for further calculation. Of course,many of these concepts are basic in the design of any algorithmic language;but they appear in ANALITIK even at the system/user interface(offering various choices of procedure) rather than as hidden consequences of system design.The of the light-pen(for selective underlining)allows calculations to be confined to certain subexpressions,for part of a computation,and restored to cover all terms,when this is convenient.The screen facility(for displaying the contents of the workspace)is useful for interactive error detection.

4.4.The fundamental operators.

The main purpose of this subsection is to outline the transformation facilities of ANALITIK--which constitute its principal means of analytical operation.Before doing this,however,I mention the other types of basic process which are used frequently.These comprise:operators for:conversion(i.e.,placing information in the working zone--e.g.,'TAKE');for naming(for storing identifiable information,to be used eventually in the working zone--e.g.,'NAME');and,for computing(to compute the values of variables--e.g.,'COMPUTE').None of these facilities differs greatly from those used in other systems,except in matters of detailed syntax,etc.;so they are not discussed further here.

The transformation facilities include the most powerful routines in the system, and,as there are(essentially)only five of them,they may be considered in turn.

The 'COMPARE' operator makes comparisons between (subexpressions of)its operand and the contents of the working zone.The operand in question may be any expression or(system of)equation(s).Comparison with subexpressions of the working zone may be made too.This is a form of pattern-matching,subsuming many of the facilities built into other systems(but requiring full implementation of any particular facility by means of a subroutine--rather than by mere invocation).Notice that the operand itself is not changed in this process;but another operation,conditional on 'comparability',may be used to effect changes.The relation of comparability is defined (rather elaborately)by means of recursion,and enumeration of special cases;in general,it is not a symmetric relation.

The second basic operator,'APPLY', uses its argument equation(s) to'operate on the contents of the working space'--roughly,by imposing on the expressions in the workspace conditions embodied in the equation(s).This idea is extended in Section 14 to cover the imposition(i.e.,assumption)of general premises and theorems,as part of the 'environment of investigation' for symbolic analysis.The intention in ANALITIK appears to be more limited,but the basic idea has the same implications. In order to accomplish this 'application',a preliminary comparison must be made,but this is only the first stage--the second being,effectively,substitution,corresponding to the equations applied,and the degree of comparability found at the first stage.

The third group of operations, for simplification, has been mentioned already, and it is best understood in the context of reduction of specific expressions. One operator closely related to those for application, and making use of various simplification routines, is the 'BRING' operator, which transforms selected expressions in the working zone to 'strong canonical form' (in effect, by repeated use of simplification rules, which are generally expressed as equations).

The fourth and fifth of the fundamental operators may be considered together, as they govern differentiation, and integration, of general or explicitly given expressions. The 'DIFFERENTIATE' and 'INTEGRATE' operators act straightforwardly on the contents of the working zone—or on selected expressions from there. Differentiation is, as usual, fully algorithmic, and is limited only by the degree of simplification attainable. The integration facility, however, doe not include a decision procedure, and integrals that the system cannot evaluate are returned in unchanged, symbolic form. On the other hand, several standard forms and 'substitutional changes of variable' (with associated table-look-up) are incorporated; and this, in conjunction with basic pattern-matching, amounts to quite a strong basic integration facility (as compared with those for other systems not using some variant of the Risch algorithm. See Section 12 for more comments). In principle, repeated integration is possible, too; but without a general algorithm this will be confined to relatively simple cases. Possibly, some future version of ANALITIK will be large enough to include a decision procedure for integration. Plainly, this would be of great value in intricate analytical problems (where formal integration is required and simple approximations are inadequate), especially in conjunction with ANALITIK's highly developed facilities for general transformation of expressions.

4.5.System operations.

These functions are similar to those in the other systems.They include operations for:control(to limit the use of algorithmic operators);for auxiliary tasks(e.g., specifying the accuracy of arithmetic operations,or provision of exact rational arithmetic);for procedures(these are similar to the ALGOL forms;they allow the repeated use of any subroutine,without its being repeatedly written out);and,for output(for interactive use,and specification of the form of output).

At present,the most effective use of MIR-2(apart from small,straightforward tasks, like forming Taylor expansions or Fourier series for relatively uncomplicated functions)is in two areas.First,in the solution of intricate but small-scale problems arising frequently(using highly optimized programs,specially developed for this purpose);and,second,for the experimental development of algorithms of some sophistication(but not requiring extensive memory)for eventual incorporation in larger-scale procedures.(See Section 12 for further comments on the mathematical scope of ANALITIK).

Although all versions of the manual are in Russian,a detailed description of the MIR-2 form of ANALITIK is given in the article by Glushkov et al.(1971;English translation,1974).Moreover,the latest form of ANALITIK(for MIR-32)is described in Glushkov et al.(1978),though this may not have been translated yet.Further information,and some new developments,may be found in the technical reports published by the Helsinki University of Technology.See,especially Korpela(1976a,b;1977a,b)and Husberg(1977).

5. CAMAL.

This system was developed(at the Cambridge Mathematical Laboratory)primarily for calculations in celestial mechanics and relativity.Later,it was extended to form a general-purpose system.CAMAL is designed as a modular system,operating at several levels.The basic level is that of polynomial manipulation;next,there is a Fourier series module;then,a module to handle complex exponentials,another for general commutative operations,and,finally,a special facility for tensor manipulations. Each of these modules has its own compiler:they are not linked during a computation,so the user must specify at the outset which module is to be used.It is more costly(in time and space)to use the general('H')than to use the Fourier('F') or exponential('E')systems,so the correct choice of module is economically as well as mathematically significant.On all levels,emphasis is put on minimizing the store used—even at the expense of increasing the time taken.This is sensible, since a program may stop for lack of space quite often,but only in exceptional circumstances will it run out of time(mounting costs being ignored in the interests of science!).However,this attitude may have been made optional in the latest version--so that it is possible to optimize programs for running time and allocation of storage. CAMAL has been implemented on IBM,PDP,CDC,and Telefunken machines.A version in the language BCPL also exists,so that,in principle,other implementations are possible,since BCPL is machine-independent.Presumably,all of the existing implementations are derived from the one in BCPL.

5.2.General sketch.

CAMAL is a self-contained,low-level language,remeniscent of the original 'Autocode'.It is a list-processing system as far as its internal structure goes,but the compiler allows input of a quasi-mathematical form.The basic elements are algebraic variables,integer variables,and atoms--all of which may be subscripted in various ways,and combined to build compound expressions.The means for constructing these compound expressions include the usual algebraic field operations, standard functions(e.g.,Elementary functions,and'user-functions',which play the part of 'arbitrary functions' in Analysis.Algebraic variables are represented by the letters A-H,and U-Z,the remaining letters,I-T,being reserved for integer variables.All of these variables may be included in 'ordered lists',if each component variable is indexed by an integer variable of suitable value.All distinctly indexed forms of the same(literal)algebraic variable are teated as independent within a given program(which ensures that one need not run out of symbols in cases where many different variables must be introduced :e.g.,A [7];A [I[4]];A[J+L[1]]). The other fundamental constituents of CAMAL expressions are the atoms,which may be viewed as'abstract indeterminates'.Atoms,too,may be subscripted,the subscrips being any integer variables(placed in angular brackets:e.g.,b < 2 >;g < J [K]+L >). Notice that all atoms are denoted by lower-case letters—so that,CAMAL is a two-case language,a fact that causes some problems,but which can be circumvented (albeit,rather clumsily and impractically).In fact,atoms may have any number of subscripts(as is essential in relativity,for instance).

Arrays of algebraic variables are denoted by A[integer expression],as mentioned already.In spite of the apparent ambiguity between,say,A[K] and the Kth element of the array,A[L],where $L \geqslant K$,there is no confusion in practice—since, one never assigns a 'value' to an array;only to its elements.The precise rules governing the use of atoms,variables and subscripts are given clearly in the manual.The most essential thing is that,at the start of any program,one must declare(i.e.,assign types and values,to)the elements in every array to be used. (The convention is that the array A[K] has K+1 elements—A[0], ...,A[K]).

The assignment of values to variables follows a fairly standard pattern,e.g., A[J]= B[K] + C[L],which allows the element(s) 'in A[J]'to be replaced by those on the right side of the'equation'.This is,of course,standard practice. To assign a value to a single algebraic variable,one could write(e.g.) A = b.2 + 7pqr (to denote the assignment $A = b^2 + 7pqr$);which indicates that raising to a power is denoted by ' . ' , and multiplication by pure juxtaposition,thoughthere are alternative notations for powers).The use of 'loops' of the 'FOR ... REPEAT' type,and of conditional statements of the form 'IF ... THEN' ,are familiar from numerical languages.Unconditional jumps are achieved by using a 'GOTO' instruction.One disadvantage of CAMAL is that subroutines do not exist(in the sense that,even when a library procedure exists for some facility,the whole procedure must be written into the CAMAL program).A similar situation obtains for ANALITIK;but there is some compensation in the relative generality of the ANALITIK transformations.However,once a procedure has been written into a CAMAL program,it may be activated automatically,by using the directive '→' (conditionally,or otherwise).This means that each procedure must be labelled(and the rules for consistent labelling seem very obscure!).Otherwise,the only labels needed are used to identify segments of the program—for (un)conditional jumps.Conditional statements governed by order relations among (real)variables,make use of the usual symbols for order on the Real Line.Subroutines can be invoked unconditionally(if they are already in the program)by sandwiching the label of the subroutine between two'arrows',e.g., → I[K]→ .

5.3.Outline of general procedures.

Formal differentiation,and other operations on general functions may be done by defining the user functions to show explicitly the variables on which they depend. For instance, A = f [x,y] ; B = d(dA/dy)/dx ;PRINT [B] ,produces a second-order, mixed partial derivative of f.However,the 'short cut',A = d(df/dy)/dx ,is invalid (and prompts an error message),since the system 'does not know that f is explicitly dependent on x and y'.

Integration is only nominal in CAMAL(though 'integration by parts',and various results for Elementary functions,are included as basic strategies).Factorization, and related procedures(e.g.,partial fraction decomposition)are also largely absent (and the system is geared aminly to the efficient performance of manipulations involving polynomials).

A complete program in CAMAL comprises a collection of statements and instructions (separated by semicolons,or by 'NEWLINE' directives),and finishing with the words 'STOP','END'--the first ensuring that compiling is stopped,and the second that execution of the program ceases.There are several facilities for altering the form of output(though this is always one-dimensional).Comments on the program may be inserted between colons--without disrupting the flow of control;they must be preceded by the word 'TEXT'.As in all systems,there are two classes of simplification routines:automatic and optional.The automatic facilities include:suppression of added zeros;collection of polynomial multipliers;imposition of algebraic index laws;minimization of the number of divisions(using $x^{-1}y^{-1} = (xy)^{-1}$),and imposition of the identity $fof^{-1}(x) = f^{-1}of(x) = x$,for the composition of Elementary functions and their inverses.Side relations are used,too--for instance 'SET $i^2 + 1 = 0$',ensures that i has its customary rôle as $(-1)^{1/2}$ .The optional simplifications-- all set by statements of the form 'MODE = N(an integer)'-- include use of noncommutative differentiation(e.g.,for 'unequal mixed partials'), linearization of products of trigonometrical functions,and various I/O and debugging devices.

Storage management(essentially,the use of internal simplification and reduction procedures)is covered by several facilities in CAMAL.No expression is duplicated, and a 'use-count'facility ensures that expressions that are not in 'sufficiently regular use'are removed from the store.Other important facilities include 'SORT' (which collects similar terms);'SUB',and 'FSUB'(which substitute any suitable expression for,respectively,an algebraic variable and a function);'EXPAND'(which can assign terms of different orders,in its polynomial argument,to different storage locations;'LOSE',and 'CLEAR'(which remove from store,respectively,an algebraic variable and an array of such variables).The setting 'MAXORDER = k',ensures that all terms in an expansion of order k+1 or more are discarded.The assignment of weights to variables,using 'WEIGHT', allows all terms of the same weight to be collected;and then,if possible simplified.Other procedures for dissection and examination are 'SELECT','PARSE' and 'MASK',whose basic use and effective combination are discussed in the manual.Finally--and importantly--there is the 'colon facility',which causes store to be freed as soon as the expressions occupying it have been used.For instance, A = B: + C:; puts the 'value'of the sum of the algebraic variables B and C 'into A',after which B and C are erased.If this facility is used consistently throughout a program then much space can be saved.A difficulty is,however,that expressions may be required several times,in widely separated parts of a program;and ,every time they are removed,they must be put back later on. Keeping track of this may be very hard.(There was also a 'bug' in this facility in the 1977 version,but this may have been eliminated now).

There are very few references for CAMAL(aside from papers where it has been used in relativity and other calculations--see the references in this paper,especially, d'Inverno(1978)and Fitch(1979)).The manual(Fitch,1975 onwards)is available as computer print-out--which allows for revisions to be inserted.Collections of demonstration programs(and output)are obtainable similarly.

6. FORMAC.

The original form of FORMAC was very limited in scope—its main aim being to show that formula-manipulative systems based on FORTRAN or PL/1 could be constructed. It was limited to operations in which 'tidying up' was the dominant activity.However(as mentioned in Section 0),many of the basic problems of system design were fomulated initially in connection with FORMAC.The more recent versions(known as SHARE FORMAC)are much more powerful than the original system,and are competitive in several areas of application.The essential structure of FORMAC has not changed since 1973;indeed,the latest version is called FORMAC 73(1978 version);which underlines the fact that symbolic computation systems are developed over periods of years,rather than months.

6.2.Basic structure.

FORMAC variables are of two types:atomic and assigned—the difference being that an atomic variable has not been assigned any value(i.e.,it has not appeared on the left side of any executed FORMAC statement);hence,an atomic variable stands for itself.On the other hand,an assigned variable represents or names an expression—its current value,which will change,in general,as a program proceeds.There is no special notation distinguishing the two types of variables(in contrast to the CAMAL convention);but the default situation is that a variable is atomic unless it has been assigned a value explicitly.Again(unlike in CAMAL,ALTRAN or ANALITIK,among others),FORMAC variables—even arrays—need not be declared.Up to four subscrips may be assigned to any variable;but,presumably,this limitation, which could prove awkward in,say,relativity calculations,can be removed.
The syntax of FORMAC is essentially that of PL/1.Nevertheless,the PL/1 and FORMAC variables are kept quite separate during the running of a program—even if they have the same names.The basic method of assigning FORMAC variables(which cannot be used for PL/1 variables)is to use the 'LET'operator.Thus,A = 5;LET(A=7);B=A; results in the common value,5,for the PL/1 variables,A,B;but the value of the FORMAC variable,A,is still 7.Apart from this,the index laws,order of precedence in evaluation,etc.,are the same in FORMAC as in PL/1.
A general FORMAC variable is a string of special,numerical and literal symbols (comprising from one to eight characters,in all).Subscripts(at most four)may be represented by any real-valued FORMAC expressions(whose values may be rounded up to integers).Atoms are just 'indeterminates';but assigned variables are put into three classes,namely:LET-variables(each being assigned a single expression,which is its current value);CHAIN-variables(each being assigned a whole chain of expressions as its current value);and,STACK-variables(each being assigned a stack of expressions,the 'topmost one'at any time giving the current value).Changes of

status of variables are effected automatically,as a program runs.

In addition to these three classes of assigned variables,there are two other categories.Function variables,cover all representations of functions of any number of variables;subscripted families of functions may be included,too.The declaration 'format' is used to specify the procedure(s) to be used in any program(this is analogous to the 'procedure' statement used in ALGOL-based languages);and,sometimes,to define the syntax of the procedures.Most functional transformations can be specified in this way.The other class of variables consists of those which are neither completely free(like atoms),nor bound(like assigned variables).Rather, they may be bound,temporarily(during an evaluation)and then freed again,in what amounts to an automatic invocation of a facility analogous to the CAMAL'colon facility',thoughthe twoare not identical.(Special variables for referring to the Nth argument of a function;or to a 'subpattern of a global pattern'are also available).

6.3.Various operational procedures.

The list of 'reserved expressions' in FORMAC(combinations of letters used to denote various declarations,functions or instructions)gives a fairly good idea of the range of the current FORMAC system.All of the Elementary functions are covered, both for algebraic operations and for basic analytical procedures.There are also several functions for factorization,expansion of products,return of numerator or denominator from a rational expression,and dissection of expressions;as well as certain pattern-matching functions.However,the integration facility is weak,and this places restrictions on the type of problems for which FORMAC is suitable.In the manual, programs are given,illustrating the scope of the system in a variety of areas,and showing how the basic functions and system operations may be used efficiently.

The basic FORMAC system is for batch operation,but an interactive version does exist.The main implementations have been on IBM computers.The interface with PL/1 is explained in detail in the manual.Essentially,PL/1 features are used to structure programs into blocks or procedures,to control the flow of statements,loops and tests,to input data or parameters,and to do purely numerical computations.On the other hand,a FORMAC program is basically a PL/1 program,to which certain extra operations have been added—including the special FORMAC commands,functions,and operators active at the PL/1 level.The output in FORMAC is two-dimensional—which is especially important in the context of dialogue use,where the user reacts to the output produced at any time.

The current manual(Bahr and Knoble(1978-) is available as machine print-out;occasional FORMAC Newsletters are issued,too.

7. REDUCE.

This system was developed by A.C.Hearn(initially,at Stanford University,now,at the University of Utah,where an active research group has been formed,in the Department of Computational Physics.The original aim was to do calculations of importance in high energy physics(especially,in quantum electrodynamics).Later,it was decided to extend REDUCE to a general-purpose system,but the basic design has remained largely unchanged.The system is written in a dialect of ALGOL 60,interpreted into LISP,which is the language underlying the whole system.Indeed,REDUCE has a 'symbolic mode'(employing LISP constructions directly,and handling list-processing problems)as well as the 'algebraic mode',in which formula-manipulation is done.
The basic capabilities of REDUCE are as follows.

→ Manipulation of rational expressions;
→ formal differentiation;
→ various types of substitution and pattern-matching;
→ determination of the greatest common divisor of pairs of (n-variable)polynomials;
→ automatic and user-controlled simplification;
→ matrix manipulations; and,
→ tensor operations.

For most scientific applications,the algebraic mode is used.The user may discover which mode is in operation by typing '!*MODE' .This is important,since,there are differences in evaluation procedures in the two modes(notably,in the symbolic mode, LISP evaluation prevails,assignment statements are handled as in LISP,and,above all,sums and products are not defined at all).REDUCE may be used in both batch and dialogue forms.The I/O is two-dimensional and,as such,fairly comprhensible for rapid user-response.Moreover,anyone experienced with ALGOL will find REDUCE easy to use.During interactive use of the system,the 'PAUSE' and 'CONTINUE' facilities allow extra input(e.g.,data)to be inserted before the program progresses.

7.2.Some basic characteristics.

REDUCE is quite portable.It has been implemented on IBM,DEC PDP,UNIVAC and CDC amchines;the underlying LISP form makes it essentially machine-independent.The arithmetic can be either exact,or else arbitrary precision,floating point.It is emphasized in the manual that the system may be used on several different levels, depending on the complexity of the calculations to be done.However,these 'levels' are not syntactically distinct(unlike in CAMAL);it is,rather a question of using only limited ranges of facilities—near to minimal for particular classes of computations—and enlarging the range of procedures available(moving to a 'higher level') only when the demands of the calculation exceed the capacity of the facilities at hand.In particular,the use of the symbolic mode(which amounts to a direct use of

LISP)is classed as an activity at the highest level,and so on.This hierarchical description of the system is made clear in the manual;it is pedagogical rather than structural.

In LISP,every variable or expression must have a value(for evaluation);so,the ' operator is used in REDUCE to mirror the QUOTE of LISP—'(PQR) corresponds to (QUOTE(PQR)).Assignment statements are similar to those in ALGOL—e.g.,A:=B+C; but,as in all 'algebraic languages',B and C must be understood as any allowable, symbolic expressions,whose sum replaces the expression labelled 'A'.The default evaluation of any REDUCE expression includes basic simplification routines,and substitution from any side relation specified by using the 'LET' facility.Conditional statements of various types are allowed—typically,the 'IF ... THEN ...ELSE' construction may be used(which is especially suitable when two distinct methods of calculation may be adopted,depending on the outcome of some intermediate step). Loops are introduced with the syntax 'FOR < variable > ::= < arithmetic expression > STEP < arithmetic expression > '.Unconditional jumps are made by using the 'GOTO' instruction,in a standard way.Substitutions may be handled by using 'LET' for single expressions(e.g.,LET H(X,Y) = X**N + Y**M)or else—for substitution within arguments of functions—by using the 'SUB' facility.The syntax of REDUCE is similar to hat of FORTRAN,but with some ALGOL characteristics,too.The explicit use of the LISP forms,'CONS,CDR,CAR,QUOTE,EQ,etc.,gives REDUCE a distinctive appearance.However,the output,when the two-dimensional form is used,is comparatively easy to read,and has few unusual features;exceptthat Greek letters are written phonetically(e.g.,ALPHA),which can be awkward when,say,power series are involved. (Certain other systems,e.g.,ALTRAN,share this disadvantage;see Section 13 for more remarks).Expressions in REDUCE are of three general types:'numerical','scalar',and 'Boolean'.Strings of scalars amy be used to designate vectors,or matrices(in terms of their column or row vectors).Tensor manipulation(of the indicial type)is possible.

7.3.Structure of programs.

Roughly,a program is a sequential evaluation of functional commands,comprizing declarations,statements and expressions.These elements are represented by sequences of numbers,variables,operators(some of these types being in strings),reserved words and delimiters.As just mentioned,all Greek letters and other special symbols that are not part of the set of REDUCE characters are represented by their spelt forms.REDUCE is a procedural language,with deliberate emphasis on allowing users to create their own procedures for special purposes—which are named directly whenever a new routine is created.As a very simple example,consider the calculation of factorial N.The following construction may be used.

```
INTEGER PROCEDURE    FAC(N);
BEGIN INTEGER  M;
M = 1 S
L1: IF N = 0 THEN RETURN M
M:= M * N
N:= N-1 S
GO TO  L1
END
```

The 'dollar' sign is used to terminate all command lines.
The special REDUCE procedures include some for ordering,partitioning,defining, printing,and other output specifications.Among the ordering facilities is the LISP 'LAMBDA operator'(cf.,Section 1)which caters especially for substitutions into arguments of functions of several variables,where inherent ambiguities arise (for a machine).Another useful facility gives the LISP equivalent of any sequence of standard REDUCE input(but without evaluation).Dissection and sorting routines are also built in.This combination of special,basic facilities,and the freedom to create wide-ranging procedures makes REDUCE a powerful system.
It is worth noting that REDUCE operators are classified into two types--infix, and prefix,with the subclassification into 'assignment operators'(:=);'logical operators'(OR,AND,NOT,MEMBER);'relational operators'(EQ,and its negation,and order relations);'symbolic operators'(used only in LISP-like calculations),and 'arithmetic operators'(standard).Although all systems must,at least implicitly, admit most of these types of operators,the structure of REDUCE(e.g.,the 'levels') is organized quite closely around these distinctions among operator types.
As a second example of a REDUCE procedure--this time,using substitution and differentiation--consider the generation of the Legendre polynomials from the formula: $p_n(x) = (1/n!)(\partial/\partial y)^n \{ y^2 - 2xy + 1 \}^{-1/2}\Big|_{y=0}$ .In REDUCE,this may be rendered succinctly as:

```
ALGEBRAIC PROCEDURE  P(N,X);
SUB(Y=0,DF((Y**2 -2X*Y +1) **(-1/2),Y,N))/(FOR I:= 1:N PRODUCT I) S  .
```

This illustrates again the extreme compactness of the notation,which can lead to storage overflow unless care is taken.However,the LISP 'garbage collector' is implemented automatically in REDUCE;and it is possible to monitor the total time or store used at any point of a computation.The intercommunication of the two REDUCE modes(in the symbolic definition of functions,when they are to be used as operators on algebraic expressions)may be achieved through the declaration 'OPERATOR',followed by the description of the functions to be used.Again,the user may re-name any identifier by means of the 'DEFINE' facility.This is especially useful if shorthand symbols for long strings of variables are introduced during a calculation.

Equations may be handled as Boolean variables(of the form $E_1 = E_2$,as in ANALITIK) with values 'T' or 'NIL'.These expressions may be numbered and manipulated,so thatsubstitution from one into another is possible.Control of storage space is covered by various facilities,among them,'WTLEVEL'(for attaching weights to variables);'LET'(to impose simplifying side relations),and 'CLEAR'(to remove side relations that are no longer relevant).Selective substitution may be effected by using the 'MATCH' function(permitting substitution only subject to specified conditions).Functions for operations in linear algebra include 'DET'(to evaluate determinants);'MAT'--with syntax MAT($r_1,\dots,r_m$)--which sets up the matrix with row-vectors $r_1,\dots,r_m$;for which the individual elements may be identified by labels(e.g.,as Y(p,q),for the element in the pth row and qth column);and,'TP', and 'TRACE'(which return,respectively,the transpose of any matrix,and the trace of any square matrix).All of these operations require some preliminary 'ARRAY' declaration(similar to that in FORTRAN).Definitions required to apply to all occurrences of the arguments of functions cannot be handled by the 'LET' operator.For instance,'LET H(X,Y) = X **M *Y**K,would leave H(X,Z) unchanged.The correct form is: FOR ALL X,Y LET ... .A related operator which could be used in this case is 'ARB'(arbitrary): ARB X,Y LET ... .Again,automatic elementary integration routines may be built into the system(e.g., FOR ALL N LET X**N=X**(N+1)/(N+1);but note that repeated substitution—the default setting—must be suppressed,if chaos is to be avoided!).The 'SAVEAS' and related functions allow file-handling(e.g.,for batch use of the system).

The general facilities for practical examination of expressions(dissection,simplification,factorization...)are strong--in the sense that the user may augment the basic system functions with specially constructed procedures.This is especially so for indefinite integration,where implementations of the Risch algorithm have been made(for certain classes of integrands—see Section 12).The system is adequately(but not exhaustively)described in the manual,where details of many special facilities are given.,together with a compact'outline of REDUCE',and a list of all of the 'MODE flags',which control the settings of the many options available to users throughout a dialogue session.The manual(Hearn(1973-))is complemented by sporadic 'REDUCE Newsletters',and many technical reports(all of which, up to 1978,are listed in Newsletter No.2),mostly by members of the Utah Computational Physics Group.

8. SAC-1.

This system is developed by G.E.Collins,colleagues and research students at the University of Wisconsin,Madison.In many ways it is distinct(in aims and construction)from the other systems considered in this paper.Its closest parallel,syntactically,is probably,the original ALTRAN,as constructed from ALPAK,a collection of FORTRAN routines(see Section 3).However,ALTRAN,even in the old form,had some pretension to general use;whereas,SAC-1 is strongly biased towards creating optimal routines for calculations involving polynomials and general algebraic number fields.The whole system(including the list processing facilities)is written in FORTRAN,and all of the commands are sequences of FORTRAN calls—mainly aimed at bringing in the various FORTRAN subroutines in suitable order.Thus,SAC-1 is not in itself a 'language'(as distinct from the FORTRAN underlying it),and it can be used only with a detailed list of 'operational rules',which have few suggestive qualities,to aid the memory.A more user-directed 'METASAC' language was designed by d'Inverno and Russell-Clark(1974),which is intended to make it easier to use the system(for instance,in the standard version,even the erasure of expressions must be specially programmed).A 'SAC-2' manual is due to appear soon,and this may incorporate significant improvements in the 'user aspects' of the design.

By contrast with the apparent opacity of the SAC-1/FORTRAN links,the mathematical algorithms,many of which are original,are described in exhaustive detail in a series of technical reports.Indeed,the total SAC-1 documentation(including material on the syntax,range of procedures and running times of various routines)amounts to more than 1,000 pages.No other system is covered in such minute detail(in fact, it is often difficult to find out what the underlying algorithms are in several systems,while,in SAC-1,each algorithm is analysed in detail).All algorithms are especially cleverly designed(in line with the 'optimization' aims for polynomial-based routines),and it is claimed that many routines run in SAC-1 far more efficiently than in any other system.This claim refers mostly to problems involving algebraic numbers,factorization and the determination of zeros of polynomials.Because of this bias,SAC-1 is less well suited than most of the other systems to performing analytical calculations of a general nature.The essential facilities are there,but they are not highly developed.For instance,there is a rational function integration routine,which computes the rational part of an integral,but merely gives the intgrand of the non-rational part,making no attempt to evaluate this in terms of logarithms and other Elementary functions.On the other hand,the means by which the rational part is calculated is particularly efficient.This somewhat anomolous approach to symbolic computation is typical of SAC-1.Everything is done as efficiently as possible(in terms of 'best asymptotic algorithms'),but some things are developed very fragmentarily—quite deliberately.Of course,the finite size of computers precludes the exhaustive development of all interesting procedures;but the polarization of SAC-1 is extreme.In this sense,perhaps SAC-1 should be viewed as a(very efficient)special-purpose package.However,it is,potentially, of general scope--e.g.,a variant of the decision procedure for indefinite integration is,in principle,available--so it is probably best to classify it as is done here.One other general characteristic of SAC-1 is the unduly large number of lines of code required to implement a procedure--which makes the initial programming effort comparatively large;but this,too,may be remedied in future versions. On the plus side,there is the fact that SAC-1 is easily portable and fairly small; it can be implemented on any(large enough)machine with a FORTRAN compiler.This has led to experiments on the possible use of SAC-1 on minicomputers(Cioni/Miola(1977)).

8.2. The SAC-1 subsystems.

The system is highly modular.Each area of application corresponds to a subsystem, contained in a 'module',the various modules being interdependent.Each module is fully described in a technical report giving listings of all of its FORTRAN subprograms and analysing the algorithms used.One of the chief claims made for SAC-1 is that it is very efficient(quick,accurate and relatively cheap)for the classes of manipulations on which it concentrates.The manner in which this efficiency is achieved is explained in the reports.By making a careful choice of subsystems for a given problem,a user can minimize the storage required—storage occupied by each subsystem is listed,in numbers of 36-bit words.With the numbering of subsystems used in this paper,the interdependence of the SAC-1 modules may be summarised as follows(where (m) ρ(n) means 'the use of subsystem m requires subsystem n'). (5) ρ(4) ρ(3) ρ(2) ρ(1);(6) ρ(5);(9) ρ(5);(10) ρ(5);(11) ρ(5);(7) ρ(6); (8) ρ(6);(12) ρ(8);(13) ρ(8);(12) ρ(11).This was the position as of 1976;there may be some extra module(s) by now,with various interdependencies relative to the modules listed here--but the basic organization of the system is certainly unchanged.

The following,brief descriptions(essentially confined to explaining the scope of each module)are based on those given in the SAC-1 User's Guide(Collins and Schaller(1976)),which also contains information on running procedures,de-bugging, error messages,particular implementations,running times of all algorithms,etc.

(1).List processing(LP).

This includes all basic list processing facilities for the system,so it is required in almost all of the general algorithms,as most objects in SAC-1 have list representations.The means for invoking LP within other subsystem programs are described, and the fundamental programs for LP are listed.

(2).Integer arithmetic.(IA)
All basic arithmetic operations,I/O facilities and (numerical)calculation of greatest common divisors of pairs of integers,are covered here—in each case, for arbitrarily large(infinite precision)integers in list representation.

(3).Polynomial system.(PO)
This covers:field operations,substitution,evaluation,differentiation,and I/O— all for general polynomials in any number of variables,with infinite precision (integer) coefficients.

(4).Modular arithmetic.(MA)
These routines include:operations over Galois fields of odd(single precision)prime degree—e.g.,Chinese Remainder ~~Theorem,~~ interpolation,generation of lists of(single precision)primes in GF(p),and factorization in GF(p),where GF(p) is the Galois field in question.

(5).Polynomial GCD and resultant system.(GR)
These programs are all based on 'fast modular algorithms'for polynomials in any number of variables.They are claimed to be outstandingly efficient.

(6).Rational function system.(RF)
Most important here are:field operations for multivariate rational functions with infinite precision integer coefficients.Also covered are:differentiation, substitution,I/O,and modular algorithms for polynomial multiplication and division.

(7).Partial fraction decomposition and rational function integration system.(RI)
This provides a partial fraction decomposition(relative to a 'square-free factorization'—of importance in the general integration procedures:see Section 12)for any univariate rational function.Further,it gives the indefinite integral of any such function,in the form of the rational part plus the integrand of the non-rational part.

(8).Polynomial real zeros.(RZ)
This computes,to prescribed accuracy,the real zeros of univariate polynomials with infinite precision integer coefficients.

(9).Linear algebra system.(LA)
Here,fast modular algorithms are used to perform addition,multiplication and inversion(of matrices),null space calculation and the solution of sets of linear equations—all for matrices whose elements are multivariate polynomials over the (infinite precision)integers.

(10).<u>Polynomial factorization</u>.(PF)
This produces complete decomposition into factors irreducible over the ring of integers(for univariate polynomials only).Typically,polynomials of degree at most 25 may be factorized in less than one minute.

(11).<u>Gaussian integers and Gaussian polynomials</u>.(GP)
This includes:field operations,substitution,differentiation,I/O,and greatest common divisor calculations—all for multivariate polynomials whose coefficients are Gaussian integers.

(12).<u>Complex zeros</u>.(CZ)
This subsystem computes(to any specified precision)<u>all</u> zeros of any univariate polynomial with Gaussian rational coefficients.

(13).<u>Real algebraic numbers</u>.(RA)
This covers all of the standard operations in(respectively)the algebraic number fields,Q(α),and the algebraic function fields,Q(α) [x],where α is any real, algebraic number.Among the operations included are substitution,greatest common divisor computation,and comparison(relative to the order in Q(α)).

All of these subsystems are documented meticulously in the corresponding technical reports(including listings of all the basic FORTRAN programs).Moreover,a list of nearly six hundred algorithms(numbered,and indexed by their page numbers in particular technical reports—e.g.,LP 26,or IA 17)is given in the User's Guide. Each algorithm is also given a mnemonic describing its essential content(if correctly interpreted),and decodable by means of a further list of basic abbreviations.Several random selections from the list of algorithms indicated that(for me, at least)yet another list of codes to decode the original codes was highly desirable!In spite of this difficulty,it is plain that SAC-1 is very carefully organized.A useful article by Collins(1971)gives a brief survey of the whole SAC-1 system--including outlines of the basic mathematical algorithms used.Since the development of algebraic computation systems is so slow a process,it is unlikely that the form of these algorithms has altered significantly since 1971,even though new facilities may have been added,and details of implementations changed.Potential users are advised to read Collins' survey before deciding whether to plunge into the technical reports.The User's Guide is all right,as far as it goes;but <u>it</u> contains no examples of programs(i.e.,of input or output),though the general procedures for invoking the subsystems are given,along with declaration and initialization procedures for global variables.Brief commentson the mathematical scope of SAC-1 are made in Section 12.

SAC-1 Technical Reports

System	Report
(LP)	The SAC-1 List Processing System, by G. E. Collins. U.W. Comp. Sci. Dept. Report No. 129, July 1971, 34 pages.
(IA)	The SAC-1 Integer Arithmetic System - Version III, by G. E. Collins. U.W. Comp. Sci. Dept. Report No. 156, March 1973, 63 pages.
(PO)	The SAC-1 Polynomial System, by G. E. Collins, U.W. Comp. Sci. Dept. Report No. 115, March 1971, 66 pages.
(MA)	The SAC-1 Modular Arithmetic System, by G. E. Collins, L. E. Heindel, E. Horowitz, M. T. McClellan and D. R. Musser. U.W. Comp. Sci. Dept. Report No. 165, Nov. 1972, 50 pages.
(GR)	The SAC-1 Polynomial GCD and Resultant System, by G. E. Collins. U.W. Comp. Sci. Dept. Report No. 145, Feb. 1972, 94 pages.
(RF)	The SAC-1 Rational Function System, by G. E. Collins. U.W. Comp. Sci. Dept. Report No. 135, Sept. 1971, 31 pages.
(RI)	The SAC-1 Partial Fraction Decomposition and Rational Function Integration System, by G. E. Collins and E. Horowitz. U.W. Comp. Sci. Dept. Report No. 80, Feb. 1970, 47 pages.
(RZ)	The SAC-1 Polynomial Real Zero System, by G. E. Collins and L. E. Heindel. U.W. Comp. Sci. Dept. Report No. 93, Aug. 1970, 72 pages.
(LA)	The SAC-1 Polynomial Linear Algebra System, by G. E. Collins and M. T. McClellan. U.W. Comp. Sci. Dept. Report No. 154, April 1972, 107 pages.
(PF)	The SAC-1 Polynomial Factorization System, by G. E. Collins and D. R. Musser. U.W. Comp. Sci. Dept. Report No. 157, March 1972, 65 pages.
(GP)	The SAC-1 Gaussian Integer and Gaussian Polynomial System, by B. F. Caviness, G. E. Collins, H. I. Epstein, M. Rothstein, and S. C. Schaller. (In Preparation.)
(CZ)	Algebraic Algorithms for Computing the Complex Zeros of Guassian Polynomials, by J. R. Pinkert. U.W. Comp. Sci. Dept. Report No. 188, July 1973, 322 pages.
(RA)	Algorithms for Polynomials Over a Real Algebraic Number Field, by C. M. Rubald. U.W. Comp. Sci. Dept. Report No. 206, Jan. 1974, 224 pages.
(UG)	SAC-1 User's Guide, by G. E. Collins and S. C. Schaller. U.W. Comp. Sci. Dept. Report No. 269, Jan. 1976.

9. SCRATCHPAD.

This is the IBM 'House system'(available only to employees of IBM--apart from a few people developing the system,e.g.,at MIT,Cambridge and IAC,Rome).This is a pity;for,in terms of language construction,SCRATCHPAD is the most sophisticated and innovative of all symbolic computation packages.It runs on a specially modified IBM machine at the IBM Watson Research Center,in Yorktown Heights,NY.The system was designed by J.H.Griesmer,R.D.Jenks and D.Y.Y.Yun,who remain in charge of its maintainance and extension.Many of the algorithms and basic facilities of SCRATCHPAD are adapted from those in other systems(for instance,simplification and pattern-matching routines from REDUCE,two-dimensional I/O from MATHLAB and Moses' SIN integration package);and equally,some ideas developed for SCRATCHPAD are diffused to other systems(notably,the Risch algorithm).Thus,it is only the use of the system that is limited to members of the IBM organisation.

The crucial feature of SCRATCHPAD as a language is its declarative power,which virtually eliminates procedural programming.In other words,the user does not have to 'code'procedures(to be called and specified,as required).Instead,the minimal information needed to perform some operation is written into the system by the user--in much the same way as would be done by a mathematician doing the calculation 'by hand'.The input language uses lower case letters,numbers and special symbols;there are 'linearization rules'to render the input one-dimensional.By contrast,when output is produced,the computer 'replies' in upper case letters (with an option for one- or two-dimensional format).This makes a dialogue session strongly suggestive of a 'conversation' between the user and the system--and this is the terminology adopted by the designers.As a simple example,consider the generation of a set of polynomials from a recurrence relation.The SCRATCHPAD formulation is as follows.

```
p<0>(x) = 1
p<1>(x) = 1 + x
p<n>(x) = x *p<n-1>(x) - p<n-2>
p<n>(x)    for n in (0,1,...,11)
```

The resulting output is the first twelve of the polynomials.No other system can match the simplicity and elegance of the last instruction,which is entirely 'natural',avoiding all loops or other constructions.In this sense,SCRATCHPAD is almost ideally suited to the development of 'symbolic analysis' as it is envisaged in Section 14 of this paper.

9.2.Some basic concepts.

The standard mode of use for SCRATCHPAD may be called 'interactive,in a self-styled environment'.This environment of evaluation is tailored by the user through the imposition of 'rules of transformation'—which remain operative throughout the session,unless they are changed explicitly.Any variable may stand for:itself,a number,an expression,an array,a set,etc.(i.e.,there is no classification into types of variables—unlike in most other systems).Moreover,'unset' variables are not signalled(by default,they stand for themselves).The process of evaluation always includes simplification.The system is self-contained;but extension by users(rather than by the designers)has played a major rôle in its development.All user commands are declarative(e.g.,E = ...);the environment is represented by a 'stack' model. (In FORMAC,the concept of 'stack variables' is used,but these are related to the internal organization of a program,rather than to any set of rules applicable at the machine/user interface).Evaluation means the replacement of an'initial expression'by a'final expression',through the continuous imposition of the rules currently in the environment,until no further changes can be made.The final expression is also called the value of the initial expression.The replacement rules have the general form: 'replace L by R when C',where C includes all conditions under which the replacement is to be made.The underlying language of the system is LISP,and one facility enables users to enter a LISP mode,for certain types of computation. In this sense,SCRATCHPAD is a multilevel system.

9.3.An overview of the system..
The following comments are intended to give an indication of the general mode and scope of operation of SCRATCHPAD.

9.3.1.Certain 'ground rules'are always operative.These include precedence rules for field operations,properties of Elementary functions,power series,etc.The user-styled environment comprises a variety of'replacement rules',covering conditional replacements of initial expressions by others(as designated),until all such rules in the environment have been applied--to produce a final(evaluated) expression.Maximal evaluation within a rule(e.g.,invoking basic system functions) may be imposed by using a special operator.In all of this activity(which can also include certain pattern-matching constraints),the concept of workspace(ws) is central.Roughly,workspace is like the 'active area' in remote access systems(and, also,analogous to the ANALITIK working zone,etc.).Thus if one declares E,then the result is, ws = 'E (the ' ensuring maximal evaluation of E).Normally,all expressions to be manipulated are loaded into ws in advance--so that,a conversation cantake place uninterrupted.Past conversations stored in the system may be retrieved,if desired.In short:the environment for evaluation at any time is the stack of replacement rules imposed at that time,together with the invariant ground rules,all manipulations being done in the workspace.A SCRATCHPAD response in lower case corresponds to'arbitrary'element(s);this response may be invoked by the user, when it is convenient.

9.3.2.Various options may be set for dealing with changes in the environment during a dialogue session.For instance,one could require that the most recently imposed rule takes precedence over all others(and overrules any condition which may conflict with it);but any order of imposition can be prescribed.In any case,the command ')display rules '(which may be qualified to refer only to rules about certain objects)causes all relevant rules to be listed--the most recently imposed coming first,etc..Evaluation of a complicated expression containing unevaluated subexpressions causes automatic evaluation of these subexpressions(so that,no clumsy expression is allowed to survive through out a calculation in an unsimplified form.All evaluations of expressions are stored automatically,so that they may be used whenever a related expression is to be evaluated.These are basic facilities for storage management.

9.3.3.Overwriting or dropping of rules may be accomplished by reissuing these rules with the 'righthand parts' replaced by empty strings(e.g., f(x) = causes F(x) to be displayed,and any rule referring to f is dropped).More generally,the edict ')CLEAR rules'(possibly,with qualifiers)gets rid of all unwanted rules. Control of 'patterns'(e.g.,for standard rules related to differentiation of various types of functions)follows the same lines as for general rules.Among the routines here are matching(of the terms in two expressions),and splitting a complex-valued function into its real and imaginary parts.Patterns may be cleared at any time. Among operators useful in defining rules are thelogical operators, AND,OR, FORALL,and EXISTS('there exists'),the 'MAX'and 'MIN'operators,and,'FIND'(which returns the object(s) from a given collection satisfying prescribed conditions).

9.3.4.Flags may be set to provide options,partially determining the environment and the form of output.These flags include the following operations:additive splitting of rational expressions;reduction of expressions to 'canonical polynomial form'(one variable being 'distinguished',and the rest appearing in the coefficients,which are also,as far as possible,in canonical form.-This reduction is extended to 'forms',as well as to pure indeterminates.Thus,polynomials in,say, sin x and cosh y are treated as though the substitutions sin x = u , cosh y = v, had been made--and expressions involving indexed variables are treated as polynomials in those variables.);use of the 'chain rule'for differentiation;use of Moses' SIN integration routine(the Risch procedure is also a possible option);expansion of finite sums and products;fixing the internal ordering of variables in (function)evaluations;and,controlling the 'weight-level'of terms(e.g.,for the truncation of series expansions).

9.3.5.Patterns for certain operations may be defined and 'put into' the system (as user-designed functions).If no specific pattern is known for an operation, then a'formal operation'is indicated(e.g., df x f(x) corresponds to formal differentiation in x of an arbitrary function,f--whose dependence on x must be specified explicitly(as in CAMAL:see Section 5)).The rules for using differentiation and integration are straightforward.Formal sums may be given in several representations,the main ones being: SUM < j = i,k;n > e < j > ,

Σ < j = i,k;n >e < j >,and Σ < j in S >e < j >(each of which --with the obvious definition of S--stands for the sum of terms e_j,where j varies,in steps k,from i to i + k[(n-i)/k]).

9.3.6.Among the system functions,the following are especially noteworthy.ABS(finds the absolute value of its argument);ASYMP(for truncating power series expansions); FLOOR x and CEILING x(given,respectively,by [x],and $-$[$-$x]);COEFF(v,n,p)(finds the the coefficient of v^n in the expression p);COFAC(v,p),where v is a vector and p is a general expression(decomposes p,as far as possible,into a sum of products, with the variables of v as factors);COVEC(v,p)(forms an array of powers of v in p, with power 0 as 'origin');DEPEND(x) = b(specifies the dependence of the expression b on the 'designator',x);FACT(n)(= n!);FACTOR(p)(gives the irreducible factors of the univariate polynomial,p,over the integers);INTEGRATE(e_1,v,e_2,e_3)(finds the definite integral,with respect to v,of e_1,with limits e_2,e_3,using the SIN routine—or,possibly,the Risch algorithm);MAX(or MIN)POWER(v,p);NUMER(rational expression); REMAINDER(p_1,p_2)(returns the remainder,on division relative to a preselected main variable,for n-variable polynomials,p_1,p_2);SPLIT(e,(n_1,...,n_k))(splits formal sums or products at the points indicated—i.e.,after the first n_1 terms,then the next n_2 terms,and so on);WEIGHT(v)(used to assign a weight to variable v,to set levels of truncation for terms involving powers of v);and,SUBSTITUTE(e_1,v,e_2) (replaces v by e_1 in e_2).There are also several operations designed to handle truncated power series,including special substitution and manipulation facilities.

9.3.7.Many array facilities are provided.These include:specification(using indexed elements,and allowing nonatomic arrays,e.g.,(a,b,(q,r,s)));arithmetic operations (defined 'elementwise',e.g.,(a,b) * (c,d):= (ac,bd)),and,selection of subarrays (using 'x = x<i> for i in S).An 'ON MATRIX'setting ensures that sums of the form A + c,where A is a square matrix and c is a complex number,are interpreted as 'A + cI',Ibeing the unit matrix of appropriate order.All of the standard matrix computations are covered,too,but these are virtually the same in SCRATCHPAD as in other systems.The methods for performing the operations are not discussed,and, plainly,it is only here that significant differences are to be found.
There are also certain array operations in SCRATCHPAD that are unusual,and offer the user wide options in dissecting and 'moulding'arrays.These facilities include: CATINATE(augments a matrix with specified rows or columns'along the Kth dimension of an N-dimensional array');DIMENSION(expression)(gives the dimensions of array expressions in the form of a row-vector of integers);DROP(n,v)(removes the first n ($n>0$)or the last $|n|$ ($n<0$)elements of the vector,v.If n:=(n_1,...,n_K),then DROP operates separately relative to each of the K dimensions.);LENGTH(v)(:= the number of 'top-level'components in v);REVERSE,ROTATE,and RESHAPE(respectively,reverse the order of the elements in an array—one'dimension'at a time,effect cyclic replacements by a prescribed number of places,and,'give the second argument the shape of the first':see the manual for details);TAKE(n,v)(same as DROP,except

that elements are selected,rather than removed);SOLVE(m_1,m_2)(solves the system of n equations in n unknowns,with coefficient matrix,m_1,and 'righthand side',m_2); and,SOLVE(p,v)(a slightly more general facility,for solving any compatible system of equations in specified variables).Communication of SCRATCHPAD with FORTRAN is possible,for numerical phases of computation.

There are many other procedures,providing,together,powerful facilities for symbolic manipulation.The performance of SCRATCHPAD in standard butlarge-scale computationshas not been examined(in relation to its effectiveness in basic calculations,time taken,maximal size of problems solved)in any general publication,and it would be interesting to know how it compares with,say,SAC-1,in such areas as zero-determination.Of the elegance of construction of SCRATCHPAD,there can be no doubt: it is really a question of how far elegance is compatible with efficiency,especially in mundane analaytical computations.The SCRATCHPAD manual(Griesmer,Jenks and Yun(1975))reflects the care with which the entire system has been designed.It is brief;yet all essential information is there,with extra comments and examples for the more intricate facilities.A set of 'SCRATCHPAD conversations',illustrating some of the potential uses of the system,has been issued ;and there are some program examples in the manual,too.Other,relatively brief accounts may be found in Griesmer and Jenks(1971),and Jenks(1974).Current developments are covered in the ACM SIGSAM Bulletin(where all symbolic computation systems and new algorithms are discussed),and in occasional technical newsletters.

10. SYMBAL.

The SYMBAL system was among the earliest of symbolic computation packages.It has been developed by Engeli(essentially,in two stages,(1966,1969),and,(1975)),as a general-purpose system,with particulat attention to aspects of language design, allowing programs to be written concisely,with minimal declarations,virtually no machine-dependent operations,and few auxiliary computations.The first full description(Engeli(1969)) embodies all of the essential characteristics of the language (even,in its present form).Although the need for a dialogue mode was recognized, right from the start,it does not seem to have been provided in the latest version (available on IBM,CDC and ICL machines).SYMBAL is designed to be totally machine-independent—even word-lengths are not mentioned explicitly in the specification. All arithmetic is infinite precision(rational).As a language,SYMBAL is based strongly on ALGOL,but it has several additional structural features,for handling abstract manipulations.The full syntax is specified in 'Backus normal form',including all basic symbols and syntactical elements(unsigned integers,identifiers,simple variables, general variables,and functions),and the syntactical rules governing expressions, vectors,statements,and the structuring of programs into 'blocks'.Certain standard ALGOL symbols are not used in SYMBAL;while,there are extra symbols adjoined to ALGOL,to cover operations on symbolic expressions.

The execution of SYMBAL is interpretive,so it is not necessary to declare the types of variables(of which there are seven:'undefined','algebraic','vector','logical', 'label','string',and 'procedure'—this nomenclature being,more or less,self-explanatory).The status of a variable in SYMBAL may be 'atomic'(if it has not been assigned a value,or,if it stands for itself),or,'nonatomic'(otherwise).The type of a variable is determined completely by its value.New variables may be 'created',during a computation,by using the **new** facility(e.g., '**new** a,b,q ').As a result of value-assignments during a program,the type and status of variables can change frequently—but the system keeps track of these changes automatically.Most of the conventions used in the formation of expressions are similar to those in ALGOL(including the use of '$\uparrow$' for exponentiation,and,of the logical operators,'$\neg$'(not), '$\wedge$'(and),and,'$\vee$'(or)),and,Greek letters(and other symbols not in the basic set) appear in spelt forms:'omega',etc..Moreover,the handling of conditional statements (of crucial importance in symbolic computation) follows the ALGOL '**if** ... **then** ... **else** ' scheme.The main algebraic/analytical operations are:substitution(with syntax **S** V1:E1,...,Vn:En $\perp$ ('$\perp$' being a separator,used in listing programs), for the replacement of the variables,V_i,by expressions,E_i;differentiation (with syntax **D** V1:E1,...,Vn:En $\perp$,for ' $\Pi(\partial/\partial V_i)^{E_i}$ ';and the formation of(finite) sums and products of expressions(using,respectively, **for** i:= E1:E2 **sum** E3 , and, **for** i:= E1:E2 **multiply** E3).Here,E1,E2 and E3,are numerically-valued expres-

sions(rounded to the nearest integer,in the above constructions,where '<u>for</u> ' has its invariable syntax,namely, '<u>for</u> E1:E2,E3 ').The sum and product facilities are used to define various functions and expansions(e.g.,the first n terms of the Taylor series for 'cos x': <u>for</u> i:= 1:n,2 <u>sum</u> (-1) ↑ (i/2)*x ↑ i/fact(i); or,the first m terms of the Taylor series for '$e^{\cos x}$':
<u>exp</u>:= (t:=1) + (<u>for</u> k=1:m <u>sum</u>(t:= t cos/k)) ,where 'cos' refers to the truncated Taylor expansion for 'cos x';so that,on rearrangement,a truncated power series of order m(2n-2),in x,is obtained.The following SYMBAL expression corresponds to the determination of part of the Taylor series for a function of two variables,x,y, about the point (x_0,y_0),with 'increments',p,q.

<u>for</u> i:=0:n <u>sum</u> for k:=0:i <u>sum</u> p ↑ k/fact(k)*q ↑ (i-k)/fact(i-k)*<u>S</u> x:x zero,y:y zero ⊥ <u>D</u> x:k,y:i-k ⊥ f

This illustrates the compactness of SYMBAL expressions,as well as displaying several of the basic symbolic operations.It compares favourably with comparable representations in other systems(though,the most 'natural' mode of expression is that in SCRATCHPAD).All Elementary functions,and their inverses,are covered,and may be used in constructions such as the one just given;but,formal <u>integration</u> is minimal—a serious limitation.
Values of the 'entries' in vectors or matrices may be assigned simultaneously with the definition of these objects.The generation of vector components(or,of matrix elements),suitably ordered,is handled using a tree structure.The subscript of the 'leftmost component'is specified.Thereafter,components are 'created',either with values(of arbitrary type),or else,as atomic variables(of type 'undefined').To allow maximal flexibility,the vectors are not simply 'ordered collections of components,with subscripts 1,...,N—for any positive integer,N'.Instead,the components may be introduced in 'blocks, $[E_1^i,\ldots,E_{K_i}^i]$,of lengths,K_i,with 'subscripts' S^i+j— subject to the consistency conditions $S^i+K_i < S^{i+1}$ $(0 \leqslant i \leqslant n-1)$,the S^j being real-valued expressions.Matrices with general subscripts may be defined analogously. This generality may be useful in some calculations where it is convenient to regard as the components of a vector(or,the elements of a matrix)entities which are not associated in any direct way with finite sets of integers.Successive entries in vectors or matrices may be filled by using the <u>for</u> clause.For instance:

(i) { <u>for</u> i:= p:n+p-1 <u>do</u> i ↑ k } ,produces an n-vector with i th component $(n+p-1)^k$;

(ii) { <u>for</u> i:= 1:n <u>do</u> { 1;m }} ,produces an (mxn)-matrix of atomic variables(since, no values are assigned);and,(iii)construction (ii),with m=n,yields an upper-triangular matrix.Again,<u>matrix multiplication</u> is covered by the construction:

(iv) { <u>for</u> i:= 1:n <u>do</u> { <u>for</u> k:= 1:n <u>do</u> { <u>for</u> j:= 1:n <u>sum</u> a[i,j]*b[j,k] }}} .

There are various standard functions (as in all systems),including:binomial(n,k), delta(i,k),abs(z),denominator(z), splitpowers(P,x),round(p),and vector(z)--most of which require no explanation.'vector(z)' is a logical function to test whether its argument is of vector type(there are,also functions for finding the maximum and minimum subscripts in a vector argument).The function 'splitpowers',used in the form splitpowers(P,x),where P is a general expression,and x is a variable, has for its value a vector,whose components are the coefficients of powers of x occurring in P(if P is a polynomial in x);or,a vector containing the exponents of x occurring in P(if P is a more general expression in x).If P contains expressions in x with infinitely many different x-exponents,then,some cut-off device must be used--analogous to 'MAXORDER',in CAMAL,and to similar facilities in other systems. (In SYMBAL,a 'mode' is set to effect such truncations).

Much use is made,in SYMBAL,of definitions involving recursion.This is just one example of a procedure ,in which a 'quotation' is assigned to a variable.A general quotation comprises 'parameters','declarations','statements',and,'value-assignments'. Procedure calls automatically invoke evaluation(quotations are assigned in an executable manner).For instance,the call,'z:= Ackermann(p,q)',assigns to z the value of the Ackermann function'at (p,q)'--this function being defined recursively,by:

```
Ackermann:= formal m,n;
         := if m = 0 then  n+1
        else if n = 0 then Ackermann(m-1,1)
        else Ackermann(m-1,Ackermann(m,n-1)).
```

Some knowledge of the Elementary functions and their inverses is built into the system(e.g.,derivatives,special values,symmetry relations,and 'addition theorems'). The detailed performance of calculations is controlled through the choice of modes, which may:impose basic simplification rules(e.g.,by applying distributive laws, delaying assignments of values,and truncating power series);or,allow the output to be monitored during a computation;or,prescribe the maximum line-width in print-out, etc..All of this is much as for other systems.In the second design phase,SYMBAL was revised generally;but,the fundamental structure,and mathematical aims,remain unchanged,as formulated in the original version.Most of the innovations result in greater convenience for the user(an aim that is given high priority in the design of the system).The modes may be varied by users,but there are 'default settings', corresponding to those situations judged to occur most frequently.The formatting of output is automatic(once options have been set),and aims at coherence and readability. However,only one-dimensional output is obtainable.
A few examples of SYMBAL programs(reproduced from Engeli(1969)) will give an idea of its scope,and,of the compactness of well-written programs.The minimality of formal integration,nontrivial factorization,etc.,places severe limitations on the use of SYMBAL in analytical problems(in comparison with,e.g.,REDUCE,or MACSYMA);but,high efficiency is claimed for SYMBAL,in all areas where it can handle the calculations.

Within its analytical limitations,SYMBAL is compact,elegant and effective(though, the lack of interactive facilities is a serious drawback,for most kinds of calculations).It is unlikely that SYMBAL will be extended sufficiently to accommodate the 'heavy machinery' required for the most sophisticated analytical applications(see, for instance,the examples outlined in Section 15);but,over a wide range of basic calculations,it is an excellent system.Apart from the papers already cited,there is a SYMBAL manual,and a collection of explanatory examples:'SYMBAL--Techniques and Examples'(also,by Engeli).Both of these publication(and all technical information about SYMBAL)are available from M.Engeli,Fides Trust Company,Zurich,Switzerland. SYMBAL is <u>not</u> freely distributed,but(unlike SCRATCHPAD,which is restricted to employees of IBM),it is offered commercially.An example of SYMBAL output is reproduce in Section 13.To conclude this section,the program examples referred to above are repoduced,without comment.

Below a procedure is presented for polynomial interpolation over a set of nonequidistant points. While the abscissas a_i must be numbers, the ordinates f_i may be given in arbitrary symbolic form. Newton interpolation with divided differences is used, and the four parameters are: a is the vector of values for an independent variable, f the vector of ordinate values, n the number of points -1, and x the symbolic variable to be used in the resulting polynomial.

The lower bound of both a and f must be zero; the upper bound then is n.

```
Newton interpolation := ∇formal a, f, n, x; new i, k;
    for i := 1 : n do for k := n : i, −1 do
        f[k] := [fk] − f[k − 1])/(a[k] − a[k − i]);
    := (k := 1) * f[0] + (for i := 1 : n sum
        (k := k * (x − a[i − 1])) * f[i])∇
```

Taylor series expansions of the elementary functions occur frequently, and here we shall give a small selection of them:

```
exp     := for i := 0 : n sum x ↑ i/fact(i);
cos     := (t := 1) + (for i := 2 : n, 2 sum (t := −t * x ↑ 2/i/
             (i − 1)));
i       := sqrt(−1);
sin     := (S x : i * x ⊥ exp − S x : −i * x ⊥ exp)/(2 * i);
s       := {0: n:};
tan 1   := for i := 1 : n, 2 sum (s[i] := (−1) ↑ ((i − 1)/2)/fact(i)
                 −(for j := 2 : i − 1, 2 sum (−1) ↑ (j/2)/fact(j)
                 * s[i − j])) * x ↑ i;
R       := tan(x);
tan 2   := for i := 1 : n sum S x : 0 ⊥ (R := D x ⊥ R/i) * x ↑ i;
expsin  := (t := 1) + (for i := 1 : n sum (t := t * sin/i));
```

The program presented next applies to differential equations of any order n, and yields the first n derivatives at $x = x_0$. The vector u holds all the derivatives, while the subscripts of y denote the order of the derivative of y with respect to x. In the definition of the vector u, $m - 1$ initial conditions, which may be symbolic, and the differential equation are filled in. This program is applied to the first-order differential equation $y' = x^2 + y^2$ with the initial condition $y(0) = c$.

```
begin
    n := 16;
    m := 1;
    y := {0: n:};
    u := {0: c, x ↑ 2 + y[0] ↑ 2, n:};
    for i := m + 1 : n do u[i] := D x ⊥ u[i − 1]
        +(for k := 0 : i − 2 sum y[k + 1] * D y[k] ⊥ u[i − 1];
                                                  mode[2] := 1;
    x := 0;
    for i := 0 : n do y[i] := u[i] * 1;
end
```

SYMBAL programs for interpolation; Elementary functions, and ordinary differential equations. (From Engeli(1969)).

11.Special packages.

In this brief section,a few packages are mentioned that have been designed especially for various classes of calculations.The areas of application include relativity,plasticity,high energy physics and (finite)group theory.

The first 'purpose-built' systems were intended for work in relativity(for instance, determination of the gravitational field equations corresponding to a given metric tensor;classification of metrics into 'Petrov types';and,miscellaneous perturbation procedures).The reviews of Barton and Fitch(1972a,b),Cohen et al.(1976), and d'Inverno(1978)give a fairly complete picture of how this area of research has developed.Recent investigations correspond to the differential geometric view of relativity(see e.g.,Misner et al.(1973),and Pavelle(1977)).There were also systems designed to handle calculations in celestial mechanics,e.g.,processing of Poisson series(see,for instance,Jefferys(1970,1971),and Rom(1971)).

11.1. Relativity.

The LAM(LISP Algebraic Manipulator)system was constructed in LISP by d'Inverno.A modification(written in Atlas assembly language)was called ALAM(d'Inverno(1969)). Although it is structurally unsophisticated,ALAM is claimed to be very efficient. All arithmetic is rational;simplification amounts to erasure of expressions that are identically zero,removal of all parentheses and collection of 'like terms'.Substitution and imposition of side relations are possible,too.Extra facilities include:easy transfer of expressions to and from the store,for use in 'main calculations',and a range of operations for truncated power series.An adaptation of ALAM to CDC 6600 machines('CLAM') was made by d'Inverno and Russell-Clark(1971), using a far simpler input language than LISP(which was used in all earlier versions).An IBM 360/370 adaptation has been constructed in Stockholm,by Frick,who is also the designer of the latest system in this family,SHEEP(Frick(1977)),for use on DEC PDP machines.Unlike all of its predecessors,SHEEP is interactive,and contains several improvements to the algorithms in LAM,as well as additional facilities(e.g.,a routine for functional differentiation,and instructions for communicating with REDUCE--so that,all of the factorization and integration routines of REDUCE may be used in a SHEEP program).There are also SHEEP-based facilities for tensor manipulation(Hornfeldt(1979),and references given there);and,presumably, some routines for handling differential forms willbe incorporated(if this has not been done already).Of course,the large,general systems can cope with all of the relativity calculations;but,throughconstant modification and improvement, SHEEP has become one of the leading systems in this area.The manual(Frick(1977)) offers potential users all of the essential information,including accounts of the subpackages for dealing with particular classes of relativity calculations.

11.2.Plasticity

Another special system with potentially wide application has also been designed in Stockholm,for handling calculations in plasticity(Mouton(1979a,b),Mouton and Aman(1979)).This system,called PLAST 3,may be used in conjunction with certain numerically-orientated programs dealing with stress distributions,slip-line fields and general three-dimensional analysis.Each of these programs is fully documented, and complete input and output descriptions are available.

PLAST 3 is a symbolic manipulation system,which can communicate with both SHEEP and REDUCE.Its individual contribution lies in the provision of special subsystems (or modules)for specific types of calculations.For instance,the tensor facilities include routines for calculations in various curvilinear co-ordinate frames,and several procedures involving the basic equations of plasticity theory,at different levels of approximation.A FORTRAN-compatible output can be obtained if the program is transferred to REDUCE--and this also allows nontrivial integrations,which occur when curvilinear co-ordinates are used.It appears to be the aim to add modules covering each well-defined area of plasticity theory,with a link to the numerical programs,for applications.In Mouton(1979b),a combined numerical/symbolic procedure is followed to study metal-forming problems,and this close combination of numerical and manipulative facilities is likely to remain dominant for work in plasticity.Extensions to cover rheological problems do not seem to have been made so far,so there is wide scope for further development.

11.3.High energy physics.

Two more special-purpose packages(well-known to workers in high energy physics, but seldom mentioned outside this context)are SCHOONSHIP(based at CERN,Geneva),and ASHMEDAI(based at the Stanford Linear Accelerator Center).Each of these systems is a large collection of symbolic/numerical routines,constantly growing,as new problems arise,but,as 'languages',remaining essentially unchanged.ASHMEDAI is FORTRAN-driven,and can be used on any sufficiently large machine with a FORTRAN compiler. SCHOONSHIP,on the other hand,is not driven by any standard language,and runs only on CDC machines.Both systems are designed to deal with large-scale calculations involving noncommuting variables,in connection with elementary particles,nuclear physics and closely related fields.Full details of these packages are available from Stanford and CERN(but see,also,Strubbe(1974)and Perisho(1975)).

11.4.Finite groups.

A system quite different from any other mentioned in this secion(and,from any of the general systems)is the GROUP package developed by Cannon,and others,in Sydney, and currently maintained by them,and by Neubüser and co-workers,in Aachen.GROUP, a comprehensive collection of routines,is related to the group theory language CAYLEY in a way analogous to the relation of ALTRAN to ALPAK(in the original form of ALTRAN--see Section 3).Both batch and dialogue modes are available.The system is written in FORTRAN,and runs on CDC 6000 and CYBER machines.Although it comprised about 50,000 lines of FORTRAN in 1976(and so,is probably larger by now) CAYLEY/GROUP is said to be fairly portable.

The range of GROUP is wide;and,as in other systems,new facilities are added as they become available.A formal description of CAYLEY is given in Cannon(1976a).An earlier outline of the project is Cannon(1973),and a full discussion of the routines in GROUP may be found in Cannon(1976b).These cover:investigation of the subgroup structure of a given group(by several methods);study of permutation groups and coset tables;topics in combinatorial group theory,including the construction of defining relations;automorphism groups;character tables;mappings of groups;and,the definition of various particular groups,rings and fields.

As a language,CAYLEY is strongly based on FORTRAN,but it does have some novel features,e.g.,for distinguishing among the various types of algebraic structures (groups,semigroups,rings,fields,modules)that are met frequently in calculations involving groups.Moreover,the system is valuable both as a research tool(e.g., for generating,and possibly verifying,conjectures),and as an aid to learning about nontrivial group-theoretic results.From the point of view taken in this paper,CAYLEY/GROUP is important,since it incorporates many constructive routines which may be used in conjunction with analytical procedures to broaden the mathematical scope of symbolic computation.An extensive,computer-based bibliography on the use of computers in group theory is maintained by Volkmar Felsch(University of Aachen);it is available as machine print-out.The bibliography covers all of the algorithms(and implementations)currently in GROUP,and many others which may be suitable for inclusion eventually.Apart from the papers of Cannon already referred to,there is a manual(available from the University of Sydney).

12.Comments on scope,and on some fundamental algorithms.

In the following considerations,two facets of design are of primary relevance: capacity for user-modification,and possession of built-in routines.Of the main systems,MACSYMA probably has the most built-in facilities(and is the hardest to modify fundamentally),while REDUCE is probably the most easily modified,even on quite a basic level(but has comparatively few built-in routines).Between these 'extremes',one finds various mixtures of these characteristics.(ANALITIK,because of its hardware implementation,will be discussed,briefly,on its own;but it has very few self-contained procedures and,in this sense,is analogous to REDUCE).The essential point is that there are several 'language levels'in any system.Modifications of procedures written in the'top-level mode' are fairly straightforward (even in MACSYMA)and may be made by sufficiently experienced users,with suitable access to the system.Indeed,this is one of the main sources of improved routines, and of new facilities.However,the most fundamental features of systems cannot be altered so easily(and,for MACSYMA,such changes require much work,and an intimate knowledge of the logical design of the system).

Another consideration is the quality of the mathematical algorithms incorporated. As usual,there is no 'absolute ordering' of systems in this connection,but some are definitely more mathematically-orientated than others(which emphasize manipulative routines,producing well-simplified output,but of limited mathematical scope).In this sense,ALTRAN CAMAL,FORMAC and SYMBAL could be said to have less potential for nontrivial mathematics than ANALITIK,MACSYMA,REDUCE and SCRATCHPAD. SAC-1,is hybrid,since its basic facilities are not analytically strong,but its routines for dealing with,say,algebraic numbers,are highly developed.Again,CAMAL is said by its designers to be intended primarily for polynomial manipulations (and to be of little use if'genuinely rational'functions are essentially involved in a computation);while other systems place emphasis on the efficient handling of rational expressions(and discourage the use of series,limits or 'infinite integrals').ALTRAN,FORMAC and SYMBAL come in this category.None of these tendencies, however,should be interpreted rigidly:there are certainly overlaps in the capabilities of systems that are ostensibly in different categories.Above all,the effective value of a symbolic computation system depends critically on the quality of response(even,of anticipation)of the user,as well as various facets of hardware and software.For this reason,the most sensible policy in this section is to confine attention to a few basic algorithms(most of which have been either developed from scratch,or else,greatly extended,specially for symbolic computing)and to comment parenthetically on the capacities of systems for using these algorithms. If any sort of detailed comparison is desired,readersshould refer to back numbers of the ACM SIGSAM Bulletin,where various 'test problems'are solved on different systems,to allow direct comparisons;but,even these tests are never conclusive.

Basically,there are three types of algorithms that play a key rôle in symbolic computation.These cover:calculation of greatest common divisors(GCD's)of polynomials in several variables;factorization of polynomials over algebraic number fields;and,symbolic integration of 'arbitrary' expressions.On further examination, these procedures are found to be closely connected (in the sense that efficient integration repeatedly requires both of the other facilities—for polynomials in several variables;while, factorization and GCD often have common ground,too). This is hardly surprising,since any systematic integration procedure must work by decomposing the integrand into components of 'simpler structure',for which a decision procedure(including,where appropriate,effective methods)is known.

12.2.Greatest common divisor(GCD) and factorization algorithms.

Direct extensions of the Euclidean algorithm for finding the GCD of two polynomials in a single variable produce explosive storage demands(see,e.g.,Knuth(1969)). Since ordinary rational function arithmetic entails the removal of common factors, and,equally,GCD calculations may be used as one step in factorization schemes,the importance of efficient,multivariable GCD routines is evident.Although the systematic removal of the (numerical)GCD of the coefficients at each stage of Euclidean type algorithms does reduce the growth of coefficients at intermediate stages,it requires the performance of many numerical GCD calculations.Further refinements, due to Collins(1966,1967)and Brown(1971)still cannot prevent unmanageable growth of coefficients,in unfavourable cases.Hence,even with the improvements,this form of the algorithm is used only in relatively small-scale calculations.

An approach more in the spirit of numerical analysis also originating largely in work by Brown(1971),and Collins(1966,1967),involves interpolation,with respect to all of the auxiliary variables(for which integer values are substituted),using the Newton or Lagrange formulae—the GCD calculation for the <u>main</u> variable being performed using arithmetic modulo suitably chosen primes,which automatically prevents coefficient growth.If this scheme is followed for several primes,then a Chinese Remainder Algorithm(see,for instance,Lipson(19)),may be used to determine the GCD to 'compound modulus',which(if it is large enough—or,of high enough degree,if theGCD is a polynomial) allows the unrestricted GCD to be found.It may be shown that,for polynomials in k variables,the probability that a prime,p,is'unsuitable', in the above procedure,is at most k/p—which is acceptably small in many practical situations(where p can occupy almost a whole machine word).

Another method,having much in common with the modular/Chinese Remainder Theorem approach,is based on a constructive form of Hensel's Lemma,in which, given a congruence, $f(x) \equiv g(x)h(x) \pmod{p}$,sequneces, $\{g_k\}$, $\{h_k\}$,of polynomials,may be costructed to satisfy the conditions $f(x) \equiv g_k(x)h_k(x) \pmod{p^k}$,for $k = 2,3,\ldots$ .Thus,when k is large enough,the exact factorization is obtained(see, e.g.,Van der Waerden(1949)).An extension of this scheme,due to Zassenhaus(1969), replaces the modulus p^k by p^{2^k} .Both methods use the 'modular images' of polynomials;the essential difference between them isthat the modular/Chinese Remainder Theorem uses several primes,while the other procedure uses only one prime,avoids the Chinese Remainder Theorem altogether,but replaces it with a Hensel type construction.Since the GCD,say, δ,of f and g,satisfies the conditions $f = (f/\delta)\delta$, $g = (g/\delta)\delta$,it is reasonable to suppose that there is a natural connection between GCD and factorization algorithms;and such a connection was in fact found by Moses and Yun(1973)--leading to the so-called EZGCD procedure.A comparison of the modular/Chinese Remainder Theorem,and Hensel/Zassenhaus methods for factorization and GCD calculation has been made by Miola and Yun(1974),who list various cases in which one particular type of algorithm is preferable(one basic criterion being whether the polynomials are 'dense'--all coefficients nonzero--or relatively 'sparse').This method may be combined with 'interpolation in the auxiliary variables(at suitable integer values)to cover the multivariate cases of factorization and GCD calculation.Most of these developments have been incorporated in many of the current versions of symbolic computation systems,either as standard procedures,or else,as known procedures that can be written into any program as required.ALTRAN,SAC-1, MACSYMA and SCRATCHPAD are strong in this area(with built-in routines),and efficient implementations exist in REDUCE.CAMAL,. and FORMAC and SYMBAL appear to offer only simple factorization facilities(e.g.,removal of 'obvious' common factors);but the scope may have increased(e.g., recent facilities in FORMAC seem to cover GCD's).. Closely related to these ideas is the problem of factorization of multivariate polynomials over(arbitrary)algebraic number fields.Once again,interpolation in auxiliary variables(at suitably chosen integer values)may be used.If a 'small enough' prime,p,exists,such that the 'minimum polynomial' defining the algebraic number field is irreducible modulo p,the both the modular/Chinese Remainder Theorem and the Hensel methods may be extended to cover this class of problems.When no acceptably small prime ,p,can be found,the required univariate factorization may be effected using a two-stage procedure.First(letting F denote the algebraic number field),the multivariate polynomial over F is transformed into a univariate polynomial over the rationals(but of much higher degree);after which,this univariate polynomial is resolved into factors.The final(multivariate)factorization is obtained,in the second stage,by using a 'p-adic interpolation' method,due to Wang and Rothschild(1975).Examples and further explanation may be found in Wang(1976), and in a brief review article by Moses(1974). See also,Hearn(1979),Zippel(1979).

12.3.Symbolic integration.

The third basic algorithm(which has been developed almost entirely for use in symbolic computation)is for the integration of general,univariate functions.In spite of some outstanding problems,this represents perhaps the greatest achievement so far in symbolic computation--having,as it does,so many applications.However,as remarked already,efficient integration depends strongly on factorization,and the implementation of integration algorithms makes full use of the methods described in Section 12.2.Even in the simplest cases,for rational integrands,extensive factorization facilities,and a genral routine for reduction to partial fractions are required.In this form,an integrationprocedure(though,not a 'practical algorithm') was developed by Hermite,who showed that,in principle,integration in terms of the rational and Elementary functions is always possible'in finite terms'.

Much earlier,Liouville(1833a,b;35,37,39,40),asked,more interestingly: whether one could prove,for large classes of integrands,that the corresponding indefinite integrals were expressible as finite combinations of prescribed 'standard'functions --and that various other classes of integrands allowed no such representation.This was the starting point for the work which culminates in Risch's decision procedure and its recent extensions and implementations.Initially,several improvements were made in the efficiency of rational function integration(for which even MATHLAB and SYMBOLIC MATHEMATICAL LABORATORY offered facilities).See Tobey(1967), Moses(1967),Horowitz(1969,1971) and Musser(1971).Muchofthis work has been incorporated in the general algorithms,of which rational function integration forms a particular case.

The classical but remarkable work of Liouville was largely ignored until it was taken up in a series of(faulty,but suggestive)articles by Mordukhai-Boltovskoi (1906-09,1910,1913),dealing,mainly,with the 'elementary' integration of first-order differential equations(a topic to be mentioned later in this subsection).The next contributions of note came from Ostrowski(1946),and Ritt,whose book(1948)gave an up-to-date exposition of the field(including discussions of several related topics). In a later book,'Differential Algebra'(1950),Ritt introduced many of the ideas that are central to the recent work of Risch and others(Risch(1969,1970),Moses(1967, 1971),Mack(1975),Rothstein(1976),where many references may be found;Davenport (1979a,b,c),Moses and Zippel(1979),Trager(1979)).

The crucial idea, going back to Liouville, is to identify the integral as an element in a differential field extension (transcendental or algebraic) of a suitably chosen 'ground field'—essentially, the minimal field containing all functionally independent components of the integrand. This strategy entails finding methods for testing collections of functions for (often hidden) interrelationships. Certain results of this type were used by Risch(1969), and there have been many developments since then (see, e.g., Rothstein(1976), Risch(1979); also, Risch(1976), Rothstein and Caviness(1976)). To the extent that the ground field may be chosen, the definition of 'elementary' is built into the algorithm—but this is not a major consideration. Once the identification of the integral as a general field element has been made, its explicit evaluation may be accomplished by solving a set of 'compatibility equations' (obtained by comparing the derivative of the general form of the integral with the integrand, as originally given). When no finite set of equations (or else, only an inconsistent set of equations) can be found by this procedure, the conclusion is, that the integral cannot be expressed 'in finite terms' (relative to the set of functions in the ground field, and logarithmic extensions), and the computer will return the unevaluated integral form. With a suitable choice of ground field, it is possible to express the integrand as a rational expression in several (functional) indeterminates; after which various operations (e.g., partial fraction decomposition) may be used. The <u>basic result</u> is the following representation theorem due to Liouville (see, e.g., Ritt(1948), p.42): if f is a function in the ground field, F, and if the integral of f <u>does</u> have an 'elementary form', then $\int f$ necessarily has the form $g + c_1 \log r_1 + \dots + c_n \log r_n$, where g and the r_j are rational over the ground field, the c_k are constants, and the problem is reduced to fixing n, and then finding all of g, the r_j and the c_k.

The flavour of Risch's methods is well illustrated in the introduction to his original (1969) paper. If one defines a <u>monomial</u>, Θ (in terms of the differential operation in the field) to be either an exponential, or else, a logarithm, of a field element, then the '<u>pure monomial case</u>' (corresponding to a field $D[\theta] := K[z, \theta_1, \dots, \theta_{n-1}][\Theta]$ requires a decision procedure for integrals of rational functions of θ, with coefficients in D. Arguing heuristically, one may claim that, if such an integral <u>is</u> elementary, then it must have the form:

$$\int \frac{P(\theta)}{Q(\theta)} = \int \frac{P(\theta)}{A\,\Pi(\Theta-\alpha_j)^{k_j}} = \frac{R(\Theta)}{\Pi(\theta-\alpha_j)^{k_j-1}} + L + M ,$$

where $L := \Sigma\, c_p \log(\theta - \alpha_p)$, $M := \Sigma\, d_q \log \gamma_q$, the product and summations being taken over the ranges 1 to s, 1 to m and 1 to t, respectively—as determined by the given integrand. Here, Q is factorized over the algebraic closure, $\overline{D}$, of D, all $c_p, d_q \in \overline{K}$, the $\gamma_q \in \overline{K}D (:= D$, with K replaced by $\overline{K})$ and $R \in D[\Theta]$. Thus, the practical problem is to find all unknown elements in this representation (which is just a special case of Liouville's result).

The essence of Risch's procedure is to use the above representation in,conjunction with the identity $d(\int(P/Q)dz)/dz \equiv P/Q$,to obtain a relation of the form

$A R' + B R + \Sigma c_i C_i + S' E = 0$,where the prime denotes differentiation in z, A,B,the C_i and E are known polynomials in θ,the c_i are <u>un</u>known constants,and S' is an unknown element of D(corresponding to differentiation of the term M,above).The procedure is <u>effective</u> because a bound(say, μ)can be found on the degree of R in θ. On putting $R(\theta):=\Sigma a_j\theta^j$(where $j = 1,\dots,\mu$,and some of the a_j may be 0),Risch obtains a system of linear,first-order,ordinary differential equations for the a_j as functions of z—the c_i and S' being unknown 'parameters' in this system(which must be determined,eventually,from the original integrand,as given).By introducing partial fraction decompositions for both P/Q <u>and</u> $R/\Pi(\theta-\alpha_j)^{k_j}$,it is possible to ensure that a clearly decidable system of equations is generated. The validity of this whole,somewhat complex,procedure is established by induction on n,the number of <u>monomial</u> field extensions taking the field of rational functions into the minimal monomial extension field containing the integrand.If,at any stage, the system of equations is found to be inconsistent,then one concludes that the original integral was <u>non</u>elementary.Otherwise,the procedure can be completed effectively.Indeed,it was implemented in MACSYMA by Moses,in 1969,for this case of transcendental integrands(pure monomial extensions),to cover the integrals of various Elementary functions.Certain improvements and extensions of this implementation were made,by Norman and Moore(1976),in REDUCE,and by Norman in Scratchpad(also in 1976).Some clarification of Risch's scheme,and an extension to 'affine forms' with several parameters was made by C.Mack(1975).

There are,basically,two levels of nontrivial integration routines:those based on pattern-matching,substitution,and tables of known results(of which Moses' Symbolic INtegration scheme,SIN(Moses(1967))is probably the most extensive—although the facility in ANALITIK is of this type,and seems to be quite versatile),and those incorporating a decision procedure—of which Risch's is the prototype.The so-called structure theorem,used by Risch to test for functional independence among the components of integrands,was implemented in SAC-1 by Epstein(1975);and this gives SAC-1 the potential for a full implementation of the integration algorithm.A multilevel package ,combining algorithmic and pattern-matching routines,was implemented in REDUCE by Harrington(1978).Other schemes,having smaller scope,but requiring relatively little storage capacity have been designed,too(e.g.,by Stoutemyer(1975), in REDUCE).

All of the considerations so far adumbrated for the Risch algorithm apply only to the case of <u>transcendental</u> integrands(i.e.,elements of fields of the form $K[z,\theta_1,\ldots,\theta_{j-1},\theta_j]$,where each θ_q is the exponential or logarithm of an element in $K[z,\theta_1,\ldots,\theta_{q-1}]$,for $1 \leq q \leq j$, and $\theta_0 := z$).For this case,the basic algorithm and its refinements have reached a fairly stable state—most modifications corresponding to improvements in factorization,or to new methods for solving the'compatibility equations'.(In Risch's original method,a device was used to replace the set of first-order differential equations by an equivalent set of algebraic equations).By contrast,the situation for <u>algebraic</u> integrands(elements of fields generated formally in the same way as for transcendental extensions—butwith the difference that each θ_q is <u>algebraic</u> over $K[z,\theta_1,\ldots,\theta_{q-1}]$)is far from completely settled,though there has been remarkable progress recently(due,mainly,to Davenport (1979a),building on ideas of Risch(1970),who restated Liouville's theorem in terms of modern,algebraic-geometric concepts).

The basic problem has beenidentified as that of effective construction of rational functions over algebraic curves(whose form is determined by the integrand).An algorithm for this construction(which also involves finding the genus of the curve) was found by Coates(1968),and it is essentially this algorithm(in the setting given by Risch's reformulation of Liouville's theorem)that has been implemented by Davenport.In order to illustrate the nature of theseproblems,it is worth reproducing the modernized version of Liouville's theorem,so that thefundamental link with algebraicgeometry becomes apparent.(The meanings of the technical terms used here may be found,for instance,in Springer(1957)).

<u>Theorem</u>(Risch/Liouville).

Let ω be adifferential in the algebraic function field $\{K(x,y) \mid F(x,y) = 0\}$,and let $r_1,\ldots,r_j$ be a basis forthe module(over the integers)generated by the residues of ω(so that,at each K-place,P, ω has residue $\Sigma a_{iP} r_i$,with all a_{iP} integers).Let divisors d_i be given by the multiplicities a_{iP} at any place P.

<u>Then</u>: if ω is elementary,there are elements $v_0,\ldots,v_j$ in $K(x,y)$ and integers,m_1, $\ldots,m_j$ such that d_i to the power m_i is the divisor of the function v_i , and

$$d\omega = dv_0 + \Sigma(r_i/m_i)(dv_i/v_i) \text{ ;i.e., } \omega = v_0 + \Sigma(r_i/m_i)\log v_i \text{ .}$$

As it turns out,the crucial(and very deep)problem in converting this scheme into a practical decision procedure is to determine the integers,m_i—in particular,to decide when such (finite)integers <u>cannot</u> befound.Drawing on results of Manin(1958, 1963,1966),Davenport has solvedthe problem almost completely:if his procedure fails to find(some of)the m_i,this can only be because the original integral was non-elementary;otherwise,the procedure will terminate ,and produce the 'evaluated integral'.

Nontechnical outlines of various aspects of this work have been given by Moses(1971) and Norman and Davenport(1979).The connections with algebraic geometry,and the resulting algorithm,are sketched by Davenport(1979b,c),detailed results being given in Davenport(1979a).Trager(1979)has shown that considerable simplification occurs when the integrand contains,at worst,non-nested radicals.Apparently,the great majority of integrals found in existing tables(e.g.,in Erdelyi et al.(1954)) are of this type,so algorithms dealing with this class of integrands are likely to be far more efficient than those covering the general case.One other case which has yet to be tackled systematically,is that of 'mixed'(algebraic/transcendental) integrands.It appears that no obvious combination of the methods already developed is adequate,since it is difficult to 'disentangle' the algebraic from the transcendental behaviour,to produce two 'noninteracting' components.The search for such a procedure raises many interesting problems.

A number of auxiliary problems arise naturally in the context of symbolic integration.One of these is the so-called square-free factorization of polynomials(putting a typical polynomial into the form $\Pi(Q_j)^j$,where each polynomial Q_j has only simple factors).This reduction can be handled efficiently now(see,e.g.,Brown(1971)).A more difficult concept(already mentioned)is that of 'functional (in)dependence,relative to a specified ground field.The resulting structure theorems(see the references given above),which test for hidden functional relationships,often lead to significant simplification;but,even more importantly,they may invalidate the use of some procedure—for instance,of treating the components of the integrand as 'independent variables' (in factorizations,decompositions into partial fractions,etc.)as Risch(1969)does.Another awkward problem comes from the fact that,in Risch's procedure,all Elementary functions must be expressed in terms of(real,or complex) exponentials and logarithms.This unnatural representation often produces incomprehensible results.Possibly,an 'unscrambling' routine could be used to restore the results to more familiar forms,but this is unlikely to be practicable for complicated expressions.

Other problems closely related to that of symbolic integration(as studied so far) include the use of 'Special functions' as elements of the ground field(widening the definition of 'elementary'),and the solution of ordinary differential equations of the form $\psi(x,y,y^{(1)},\dots,y^{(k)}) = 0$.The use of 'higher' transcendental functions, such as the error function integral,is complicated by the fact that,whereas the behaviour of algebraic functions is dominated by the distribution of their poles, 'higher' functions may possess essential singularities;and behaviour in the neighbourhood of such singularities is hard to analyse.Further possibilities are provided by the inclusion of elliptic functions,Bessel functions,etc.(or,of hypergeometric functions,which subsume all of these types of Special functions—see,e.g., Millar(1978)).

So far,no systematic investigation has been made inthis direction,but an extension of Liouville's theorem allowingfor the inclusion of Spence funtions (defined by $S(x):=-\int_1^x (t-1)^{-1}\log t\, dt$)is given byMoses and Zippel(1979).

On the solution of differential equations,most progress has been made in the first-order case : $\phi(x,y,y^{(1)})=0$,where ϕ is algebraic(in contrast to the pure integration problem,in which it is the algebraic rather thanthe transcendental functions that cause trouble).The MACSYMA differential equation solver(initially developed by Kuipers(1973))is powerful,and can treatsome higher-order equations.A more heuristic approach,designed specifically for first-order equations,and not relying on extensive pattern-matching facilities such as MACSYMA possesses,is due to Schmidt(1976,1979).Another procedure,due to Geddes(1979),uses a form of Newton iteration.Up to now,no fully general scheme is available for higher-order equations, though a power series package(based on the Frobenius method)for MACSYMA hasbeen provided by Lafferty(1977),and a method based on the use of expansions in Chebyshev polynomials,has been constructed by Geddes(1977)in a form suitable for implementation in any symbolic computation system.(For efficient methods of treating power series in symbolic computation,see,e.g.,Norman(1975),Fateman(1977),Brent and Kung(1978);and,for Puiseux series—'fractional power series'—,of importance in the study of algebraic and algebroidal functions,see Kung and Traub(1978)).The problem of (in)definite summation (which has obvious applications both to approximate integration,and to the solution of differential equations)has been investigated by Gosper(1976,1977),by means of iterations of series rearrangements determined by 'splitting functions'(where the function $S_{k,n}$ determines,during the k th rearrangement,what fraction of the n th term of the series is to be subtracted from that term and added to the previous term).Gosper shows that,as $k\rightarrow\infty$,a series will,in general,be transformed into an expression involving two new series and two infinite products.In spite of this apparent complication,he shows,further,by means of examples,that the result usually simplifies to a single(more tractable)series--and,even to explicit closed form,in many cases.

12.5.More remarks on scope.

There are some other points about scope that are worth making.First,a system may be designed to handle one class of calculations optimally—to the detriment of calculations of other types.Thus,CAMAL is most effective when manipulating polynomials,ALTRAN and SAC-1 are intended for rational function computations,while other systems,notably,MACSYMA,REDUCE and SCRATCHPAD,allow a wider class of functions to be treated.Again,if ultimate numerical computation is in question,then the interface between the symbolic manipulation language and a high-level numerical language is of critical importance.Indeed,the Berkeley MACSYMA system,although smaller than the MIT version,is designed to produce efficient FORTRAN code.Corresponding facilities for other systems should be checked in the current editions of the manuals.

There are,of course,many basic mathematical procedures other than factorization and integration(for instance,matrix operations,and their development in linear algebra,to take only an obvious example);yet,the limitations in scope among current systems stem chiefly from their performance in these two crucial areas. Symbolic differentiation is trivially algorithmic,and was implemented long before any general system was envisaged(see Section 1).In fact,most systems have extensive facilities for matrix manipulations(including,in some cases,special packages for sparse matrices(REDUCE:useful in finite element computations),and Kronecker products and indicial tensor manipulation(MACSYMA)).This is one area where even the less analytically-directed systems perform comparatively well.

On the question of quality of algorithms,only tentative remarks may bemade reliably, since all sytems incorporate new implementations of facilities as improvements becomeavailable.However,as far as the built-in routines go,SAC-1,ALTRAN,MACSYMA and SCRATCHPAD maintain a very high level.MACSYMA and SCRATCHPAD are under constant development,and REDUCE(which is relatively easy to modify)has a large collection of algorithms contributed(and implemented)by users—which helps to keep the level of efficiency high.Again,many of the mathematical algorithms are implemented in several different systems,so their relative efficiencies depend on the effects of various basic design characteristics.As time goes by,there is likelyto be a tendency towards equalisation of performance in the most fundamental procedures — as far as is compatible with differing general design aims.This is especially so becauseof the availability of extremely small,comparatively cheap micro-processors. Some comments in this direction are made in Section 12.7.

12.6. Mathematical scope of ANALITIK.

In Section 4, an outline of the ANALITIK system, with an indication of its principal facilities, was given. The following brief remarks are intended to show how the design philosophy of ANALITIK (and its realisation) affects its potential scope in mathematical applications. The limitations due to small storage capacity and slow operating speed are certain to be greatly reduced, eventually; though, even if the current version, ANALITIK 74, is improved in these respects, it is unlikely to be competitive with the other systems; on the otherhand, its unique structure (and consequent procedural flexibility) offer many advantages. In particular, the approach adopted in this paper, to the mathematical use of symbolic computation (see Section 14) is well suited, structurally, to ANALITIK. However, the principal analytical weakness lies in the lack of a decision procedure for integration (even if some relatively complicated standard forms are included, for 'table look-up' and general pattern-matching). The crucial matter in the potential implementation of such a facility is the quality of factorization procedures realizable in the system—which, in turn, depends fairly strongly on size and speed; so it is doubtful whether ANALITIK 74 could include a Risch type decision procedure. Nevertheless, the next generation of ANALITIK systems may well be able to allow effective integration, while retaining the other features that make it attractive mathematically.

The basic capabilities of ANALITIK are listed by the designers as including: treatment of standard problems of (analytical and numerical) linear algebra; solution of nonlinear equations (mainly, by iteration); determination of extremal points of functions; integration of systemsof linear differential equations (essentially, only those with constant coefficients); and, approximate solution of nonlinear ordinary differential equations (mainly, by iterative methods), and of the 'differential equations of mathematical physics' (using Fourier's and related methods). In spite of this quite general scope, it is clear that, for standard (numerical/analytical) procedures, involving linear algebra, differential equations, etc., ANALITIK cannot compete with, say REDUCE or MACSYMA, both of which run on much faster, larger machines: its value lies, rather, in its approach to the construction of mathematical procedures. (For the record, it may be noted, however (see Korpela (1977a)), that ANALITIK, on MIR-2, solved the equation $\ddot{x} + \omega^2 x = \varepsilon f(x,\dot{x})$, with a general function, f, in 5 minutes, 48 seconds; while FORMAC, on IBM 360 (a much larger system) took 3 minutes, twelve seconds—even in the special case where $f(x,\dot{x}) = (1-x^2)\dot{x}$.).

12.7. Implementation of algorithms: sequential and parallel computation.

With the mass production of extremely small(and increasingly cheap)microprogrammed elements,the possibilities for designing highly modular systems(microprocessors) are being explored widely.The impactof these developments on symbolic computation seems not to have been investigated.In the present comments,some basic questions are identified,and their significance for symbolic computation is indicated.On a mathematical level,the fundamental problems hinge on the optimal combination of the sequentialand parallel('concurrent')modes of calculation.In relation to general implications for symbolic computation,the following matters seem worthy of study.

12.7.2. Identification of,and measurement of,parallelism.

This is a question of the automaticdetection of segments of existing algorithms (presently implemented only in sequential form)for which parallel computation is, nevertheless,the more natural(and more efficient)mode.In these terms,ones seeks useful measures of 'intrinsic parallel content',which can be applied to programs in existing libraries of subroutines.In such areas as linear algebra,much work has been done on implementing 'concurrent procedures' for use in microprocessors(see, e.g.,Heller(1978)for an extensivereview;and,for a more theoretical approach to the structure of parallel algorithms,Kung(1979),where many references may be found). Thus,the key problem here is to devise effectively applicable measures of parallelism.For one approach,see Gonzalez and Ramamoorthy(1970).

12.7.3. Capacity of existing high-level languages for parallel computation.

Methods have been developed recently(using techniques of mathematical logic)for studying this question—which has obvious practical importance.It has been found (see Jones(1977),where many references are given)that the main languages(FORTRAN, PL/1,ALGOL)are on the whole poorly suited to concurrent modes of programming.It follows that most of the symbolic computationlanguages (which are modelled,syntactically,on these high-level languages)cannot be very well atuned to the use of parallel schemes(though I know of no specific results about this).Work in this area has been done by members of the Programming Research Group at Oxford.

12.7.4.Design of speciallanguages for efficient parallel computation.
In view of the conlusion in 12.7.3,the need for languages in which the concurrent modeof computation is in some sense the optimal mode is apparent,and several studies of the array structures of parallel computers have been made(notably,by members of the Computer Science Department at Carnegie-Mellon University:see,e.g., Kung(1979),and references given there).However,the emphasis is on realizing certain algorithms in hardware(as in the ANALITIK system---see Section 4)rather than on analysing the linguistic peculiarities of concurrent computation schemes. For this aspect of the problem,see,e.g.,Hoare(1978a),where several references to language construction are given.One special language intended for the description of parallel processes is Concurrent Pascal(Brinch Hansen(1975)).

12.7.5.Use of parallel schemes in symbolic computation.
Quite apart from the need for special languages in which parallel modes of computation are 'natural',there is a need to design algorithms having high parallel content(so that there expression in the new languages is as simple as possible). It remains for such algorithms to be tested in symbolic computation.In this connection,it is notable that a parallel program/parallel data LISP machine is being constructed at the University of Utah;and this,when it is fully operational,should advance research into parallel algorithm design considerably.

12.7.6.Stochastic models of systems with intercommunicating components.
When computations are done concurrently,it is useful to have models of the 'flow of calculation' through the system,in which the transient state of each component is monitored.The manner in which successive stages of the process are triggered (various components are activated,stopped,re-activated,etc.)suggests that stochastic models may be used to describe the evolution of the whole system.These models are general enough to cover symbolic parallel computation,and may be helpful in pinpointing probable causes of 'blockages' in various classes of calculations.(See the paper by Kung,in the conference proceedings edited by Traub(1976)for a survey of this area,together with some specific models).

12.7.7.Logical descriptions of intercommunicating subsystems.
The successful design of languages adapted to parallel computation depends strongly on obtaining a sufficiently detailed logical description of a collection of intercommunicating components.This requires thedevelopment of novel logical techniques and semantical structures.See,for instance,Hoare(1978b,1979),Hoare and McKeag(1977).

12.7.8.Relative complexity of parallel and sequential algorithms.
Much work hasbeen done in this area,since it is vitally important when matters of cost are in question.In problems of linear algebra,the results are quite extensive(Heller(1978)contains many references).However,for morecomplicated algorithms,only isolated results are available--though a general theory of optimal algorithms and analytic complexity(mainly for iterative solution of nonlinear equations in Banach spaces,with various measures of 'data information')has been constructed by Traub and Wozniakowski(1980)--including a long,annotated bibliography.The task of comparing the sequential and parallel algorithms in terms of complexity remains largely open.

12.7.9.Automatic conversion of sequential programs into parallel programs.
This facility is essential if the vast store of library routines(almost all of which are implemented insequential mode)is not to be duplicated--often,with great difficulty(and waste of time)if the algorithms have tobe designed from scratch. Certain theoretical procedures to accomplish this conversion have been given(see, e.g.,Mazurkiewicz(1975));but the development of efficient methods of 'translation' for mathematical use is far from being realized.Plainly,it is a matter of paramount importance.

Taken together,the aspects of parallel computation mentioned in this subsection could have a marked effect on the design of the'next generation' of symbolic computation systems.

13. Remarks on I/O and simplification.

Most of the following remarks on I/O are implied in the descriptions of the individual systems.The purpose of this section is to make them more explicit--and to discuss briefly some of the deep mathematical problems that arise when systematic attempts are made to reduce general expressions to canonical forms.To give an idea of the range of types of I/O,and of the various forms programs can take,extracts form programs are reproduced at the end of this section.

13.1.Quality of I/O.

This is of great importance,because a symbolic manipulation system should allow the user to approximate as closely as possible the 'usual' mathematical activity. For this to be feasible,the output must be easily recognizable as mathematics,and the result of any operation on the current data should be discernible immediately, so that a suitable'response'can be evoked.These requirements imply that systems should have two-dimensional I/O,and that they should operate in a dialogue(interactive)mode.However,poorly produced two-dimensional output may be less useful than well-organized one-dimensional output;so there is room for compromise.
Of the systems discussed in this paper,FORMAC,MACSYMA,REDUCE and SCRATCHPAD have (an option for)two-dimensional output;and facilities for converting their one-dimensional input to two-dimensional form on the display screen.All other systems have purely one-dimensional input and output.Again,ANALITIK,MACSYMA,REDUCE and SCRATCHPAD are all interactive,while CAMAL and FORMAC have special dialogue implementations(though their standard versions are for batch use only);and ALTRAN,SAC-1 and SYMBAL allow only batch operation.The relative quality of output and ease of programming among the systems may be judged partially from the examples reproduced later in this section.Plainly,SAC-1 does not give high priority to the 'readability'of its input or output;but all of the other systems do aim for this.Ease of programming is enhanced if the program instructions are fairly close in appearance to the usual mathematical forms.In this respect,all systems except SAC-1 are quite easy to use,each having its syntax strongly based on that of some high-level numerical language.Specifically,FORMAC and ALTRAN are based on FORTRAN; ANALITIK,MACSYMA,REDUCE and SYMBAL all have ALGOL-like syntax,CAMAL is based on the original 'Autocode';and SCRATCHPAD is a'conversational'system,in which the language of communication has no obvious antecedent,but is very easy to 'write'.
Other criteria include:whether all variables must be 'declared'(as in ALTRAN, ANALITIK,and often in CAMAL);whether the system 'knows' the rules for handling the Elementary functions(in ALTRAN and SAC-1 ,special rules must be included to cover these functions);whether syntax errors are diagnosed during the input and construction of a program(rather than only after the program has been run); whether 'loops'are required for summation routines(e.g.,ANALITIK and SCRATCHPAD allow direct summation);and so on.Generally,when a basic facility is missing in

some package,it can be provided,with comparatively little programming effort.However,the need to insert even a small subroutine whenever a particlar construction or type of function is used is certainly a disadvantage in calculations where the construction is required frequently(and,often invarying forms,so that one cannot keep calling the same subroutine).All of the relevant information on syntax, basic declarations,quality of I/O,etc.,may be found in the latest manuals.
13.3.<u>Quality of information in the manuals</u>.
The ALTRAN manual is excellent,including several levels of description,with explanations(but almost no details of the algorithms used).For MACSYMA,the manual offers a complete description at user level,with brief outlines,and examples of all routines that were current at the time of printing,and instructions for calling these routines;there is even a glossary of programming terms.Other manuals are less comprehensive.In the case of FORMAC 73 and CAMAL,the manuals are stored on disc,and so may be updated fairly simply.Both manuals are adequate for the most basic information,but neither contains any description of library routines—insofar as these exist at all.As explained in Section 8,SAC-1 is not a 'language' in the same sense as for the other packages considered.Conseuqnently,there is no need for a manual--only,for instructions as to how the various FORTRAN subroutines of SAC-1 may be called.Each subsystem is documented meticulously in a technical report.Indeed it is only SAC-1 that gives complete information on all of the algorithms used.To get even partial information of this kind about the other systems, one must write to members of the design groups--and,even then,they may not know precisely what algorithms are in current use.Moreover,detailed timings and storage requirements are given for each of the SAC-1 facilities;and here,again,the treatment of other systems leaves much to be desired.On the other hand,although the SAC-1 User's Guide is adequate for its state purpose,it is unlikely to attract casual users,since it conveys little meaning on its own.The REDUCE manual gives an outline of the general facilities in the system,but many details are lacking(even on an operational level).Thus,it is desirable to use REDUCE frequently to get a realistic appreciation of its scope and power.In particular,there are several possible'modes',which may be used in different parts of a calculation:these are indicated,but not fully explained,in the manual.To compensate for this relative sketchiness,there are fairly regular REDUCE Newsletters(edited by A.C.Hearn,University of Utah Computational Physics Group),and frequent research reports(often containing new facilities or implementations)by members of the Utah group.
The care with which SCRATCHPAD has been designed is reflected in the manual.(There are also occasional newsletters).SCRATCHPAD exhibits an economy of notation and a coherence that it would be hard to surpass.All facilities are documented concisely;both input and output formats are very clear—and about as close to 'hand-written'mathematics as one should expect in a computer language;and the 'conversational mode' comes close to simulating human mathematical activity and communication.How-

ever,if one recalls that SAC-1,though inelegant and clumsy in I/O,is probably the most effective of all systems in certain concrete analytical procedures(e.g., determination of the zeros of polynomials),then it may be surmised that outward elegance of construction,naturalness of I/O,and high efficiency are(at least, partially)incompatible attributes.In this connection,a fundamental study of symbolic computation,on the level of languages,metalanguages and interfaces,would be valuable.Current research into the theory of computer language syntax and semantics makes a useful contribution in this direction,but it is likely that the special demands of symbolic computation will require some new concepts before significant progress is made.

There is no translation of the ANALITIK manual,but the article by Glushkov et al. (1971)gives a full description of the language construction,the various modes of transformation and evaluation,and other specialized constructions.The latest version,ANALITIK 74,is covered in another article by Glushkov et al.((1978)),which has not been translated yet.Much information on the practical use of the older form,ANALITIK on MIR-2,is contained in various technical reports issued by the computer science department of the Helsinki University of Technology.Although they are dealing with an old version of the system,almost all of what they say is still valid for the latest machines.

13.4.Mathematical problems of simplification.

This is a central problem in symbolic computation,and one that has been solved unevenly,in various contexts.All systems include routines for the most basic simplification;many of these are mentioned in the system outlines.However,without constant vigilance and resourceful use of reduction procedures,even apparently tame calculations ceate explosive storage demands(for instance,naive versions of the GCD algorithm:see Section 12.2).The simplification problem is essentially equivalent to that of finding canonical forms for classes of expressions;and of designing effective procedures for reducing general expressions to collections of canonical entities.

The formalisation of these procedures has proved to be a very difficult problem. Even if canonical forms can be defined unambiguously,they need not be unique—yet, how is one to decide which is the simplest of several forms?Such a decision may be strongly context-dependent,as is easily illustrated by examples.If one attempts to analyse the thought processes of a mathematician confronted with an unwieldy expression,it becomes evident that 'intuitive reactions'cannot be ignored.Certainly, such reactions are(somehow)derived from experience;but it is possible,often,to reject a strategy without(in any conscious way)first testing it—and it is this faculty that is so hard to formalise.(A similar situation is met in designing chessplaying computers:the intuitive rejection of all but a minute proportion of the legal moves,in a given posititicn,cannot be emulated by the machine,which must 'consider' all legal moves,even if,subsequently,it rejects many of them,after applying fixed rules).

The most remarkable general result(comparable in its depressing effect with Gödel's results on the incompleteness of formal systems)is the theorem of Richardson(1968). This states(roughly)that the problem of 'identifying zero',within a sufficiently wide class of expressions,is formally undecidable (see,e.g.,Hermes(1969),Chapter Six,for an explanation of this term).In a modification of Richardson's argument, Caviness(1970),by building on the result that the decision problem for solvability of multivariate Diophantine equations is itself recursively unsolvable(Hilbert's Tenth Problem:see Hermes(1969)for the meaning of 'unsolvable'),constructed functions,F (using standard field operations,the number,π,and the 'sine' and 'modulus' functions)such that the problem:'Is F(x) identically zero in x',is recursively unsolvable. (The solution of Hilbert's Tenth problem is due to Matijasevic(1970).Somewhat in the same spirit as Richardson's result—though having nothing directly in common with it—is the fascinating work of Jones et al.(1976),where,also starting from the solution of the 'Tenth Problem',a polynomial,$P(a,b,\ldots,z)$,of degree 25 (over the integers)in 26 variables,is constructed,the set of whose positive values coincides with the set of all primes—each of a,b,...z,ranging over the set of positive integers).

Richardson's result shows that no 'perfect' simplifier can be designed.However,no definite bound is implied on th potentially atainable efficiency of simplification procedures;it is just that they will remain inadequate in the face of sufficiently complicated expressions of certain types.Thus,the quest for more effective simplification routines is not doomed to failure,as long as the class of expressions to be handled in each routine is defined precisely.Indeed,Richardson himself provided a 'zero-equivalence algorithm'(based on successive reductions to problems of decreasing'complexity')which(see Moses et al.(1972))may be extended to include functions defined by first-order,ordinary differential equations.

In most of the systems considered here,cancellation(additive of multiplicative)is attempted,identities for Elementary functions are incorporated(usually,as side relations)and these functions are assigned canonical forms.Further,the relations $f \circ f^{-1} = \text{identity} = f^{-1} \circ f$,are imposed for all Elementary functions,and in any other context where they can be applied unambiguously.More elaborate strategies include the attachment of 'weights' to monomials(so that all terms of the same weight in any expression may be collected ,and simplified),and the use of selection and parsing routines to dissect expressions.All of these ideas are discussed,in context,in the system outlines.The canonical form problem is linked strongly with that of proving'structure theorems'(which aim to reveal all possible—algebraic or transcendental--relations among the elements of given classes of functions).In this way,automatic tests for equality of expressions may be designed,covering even situations where the equality is hidden,and far from obvious 'to the naked eye'.(Recall that,some versions of Risch's integration algorithm require a preliminary use of structure theorems,to ensure that the component functions of the integrand may be treated legitimately as 'independent variables',in partial fraction and factorization procedures).

One key problem(of special significance in relation to algebraic functions)is the canonical representation of expressions containing 'nested radicals'.Progress here has been made by Fateman(1971),Shtokhamer(1975)and Zippel(1977).See also,Caviness and Fateman(1976).In fact,the work of Caviness,and of Fateman,is concerned with un-nested radicals;but the basic ideas are of general value in studying the more involved situations.Shtokhamer defines a canonical form for expressions containing nested radicals,and gives an algorithm for reducing expressions to canonical form.His procedure is analogous to Risch's,in that successive field extensions are constructed to 'remove' radicals,until the original expression can be simplified to a polynomial in quantities which are algebraic over the field of rational functions.The corresponding algorithm is,however,very inefficient(but improvements are indicated,and,by now,the procedure may be useful in practical cases).Zippel emphasizes the concept of the 'nesting level'of a field;the main force of his results is to reduce the problem to the case of a single radical,by means of a 'de-nesting routine'.Another approach to simplification is taken by Hearn(1975), who tries to preserve the structure inherent in physical problems(as this often makes possible 'hand calculations' that otherwise would be excessively laborious). Several schemes for basic simplification are embodied in the various system designs;most of these are based on some kind of 'tree representation' of expressions).

A useful outline of work in this area has been given by Moses(1971b).Moses classifies simplification strategies into 'political types'(the radical ones imposing all of their rules invariably;the liberals leaving some room for a user's choice; and,the conservatives allowing users complete freedom to devise rules).In these terms,MACSYMA has a 'catholic' strategy(incorporating all types to some extent, and allowing users to vary the modes of simplification adopted).Similarly,SCRATCHPAD has several levels of simplification;and all of the other systems make provision,in varying degrees,for adding or removing simplification routines(by setting 'flags' for the appropriate options).Ostensibly,ultraconservative systems offer the widest choice,but the extra programming effort may be substantial,and the running time may be prohibitively long.The radical systems are the most efficient,but their inflexibility imposes undesirable limitations—compromise is essential here.

A problem closely related to that of simplification is to control the size of expressions(especially,in the form of 'intermediate expression swell',generated in the course of a calculation--for instance,of eigenvalues,where the final result may be very short,even thoughbulky expansions of determinants are involved).Although there is no general solution to this problem,all systems include some facilities for mitigating it.These facilities are typified by the SELECT,EXPAND,PARSE and MAXORDER operators in CAMAL,and by some form of regular storage inspection(for providing,at all times, maximal free storage space,so that new expressions can be accommodated without causing programs to stop on account of overflow).In all LISP-based systems,this inspection is realised through a 'garbage collection' facility,which erases,at any time,all expressions no longer required in a calculation. Other systems have their own means of storage management(though,mostly,these are not invoked automatically,as garbage collection is).Other means of simplification are properly regarded as editing operations,and details of these may be found in the manuals.Of course,possibilities for editing are far more extensive in dialogue systems than in those having only batch mode—which makes it essential for the batch systems to have comprehensive,efficient built-in simplifiers,if they are to be competitive.

This program solves certain types of ODE. Two methods are used.

```
(C6) SOLDER(EQN,U,X) :=                The routine is code-named SOLDER
   BLOCK([B,C,F,DISC,R1,R2,ALPHA,BETA],
         IF SOLDE(EQN,U,X) = FALSE THEN RETURN(FALSE),
         DISC: B^2 - 4*C, ALPHA: -B/2,
         IF DISC=0 THEN RETURN(%E^(ALPHA*X)*(A1+A2*X)),
         BETA: SQRT(DISC)/2,
         IF DISC > 0
           THEN (R1: ALPHA + BETA, R2: ALPHA - BETA,
                 RETURN(A1*%E^(R1*X) + A2*%E^(R2*X)))
           ELSE (BETA: SQRT(-1)*BETA,
                 RETURN(%E^(ALPHA*X) * (A1*COS(BETA*X)
                              + A2*SIN(BETA*X)))))$
```

All of this code is for running the routines, setting initial values, etc. The formal solution of the characteristic equation is also covered here.

```
(C7) /* AN EXAMPLE - THE METHOD OF UNDETERMINED COEFFS. FOR
        OBTAINING THE PARTICULAR SOLN. AS WELL */

   DE: 'D(Y,X,2) - 'D(Y,X) - 6*Y = SIN(X);   The ' symbol suppresses evaluation.

                  2
                 D Y   DY
(D7)             --- - -- - 6 Y = SIN(X) ------the displayed form of the ODE.
                   2   DX
                 DX

(C8) YH(X) := ''(SOLDER(%,Y,X)); ----- '' causes evaluation and maximal simpli-
                                          fication; % refers to previous expn.
                                 - 2 X        3 X
(D8)            YH(X) := A2 %E        + A1 %E     ------this defines the LHS

(C9) YP(X) := B1*SIN(X) + B2*COS(X)$       ----YP is a 'particular integral'

(C10) YG(X) := YH(X) + YP(X)$  ----YG  is the general soln.('complimentary fn').

(C11) PLUGIN: EV(DE,DIFF,EXPAND,Y=YP(X)); ----sets various switches.

(D11) B2 SIN(X) - 7 B1 SIN(X) - 7 B2 COS(X)

                - B1 COS(X) = SIN(X)

(C12) EQN1: COEFF(PLUGIN,SIN(X));
                                          compares coefficients of SIN(X),
(D12)                 B2 - 7 B1 = 1        then, of COS(X).
```

(From the MACSYMA Reference Manual(December,1977)).

```
(C13) EQN2: COEFF(PLUGIN,COS(X));

(D13)                    - 7 B2 - B1 = 0

(C14) GLOBALSOLVE: TRUE$ ------------------the default setting is
                                           FALSE;hence it must be set
(C15) SOLN: LINSOLVE([EQN1,EQN2],[B1,B2]);  to T,which causes the vari-
SOLUTION                                    ables SOLVED for to be set
                                            to the values found by sol-
                                            ving the simultaneous equns.,
                                  7         using 'LINSOLVE'(C(15)).
(E15)                      B1 : - --
                                  50

                                  1
(E16)                       B2 : --
                                  50

(D16)                       [E15, E16]

(C17) YG(X);

           7 SIN(X)   COS(X)         3 X        - 2 X
(D17)    - -------- + ------ + A1 %E     + A2 %E
              50        50

(C18) /* PLUGGING IN INITIAL CONDITIONS OF Y(0)=1---comments are prefaced
                 AND Y'(0)=0 */                     by the symbol /*

EQN1: YG(0) = 1;
                                   1
(D18)                  A2 + A1 + -- = 1
                                  50

(C19) DIFF(YG(X),X);

          SIN(X)   7 COS(X)           3 X          - 2 X
(D19)   - ------ - -------- + 3 A1 %E    - 2 A2 %E
            50        50

(C20) EQN2: EV(%,X=0) = 0;
                                         7
(D20)                - 2 A2 + 3 A1 - -- = 0
                                        50
```

```
(C21) SOLN: LINSOLVE([EQN1,EQN2],[A1,A2]);
SOLUTION

                                         21
(E21)                               A1 : --
                                         50

                                         14
(E22)                               A2 : --
                                         25
(D22)                             [E21, E22]  --- these values are stored as shown.

(C23) YG(X);
                                            3 X          - 2 X
            7 SIN(X)   COS(X)   21 %E          14 %E
(D23)     - -------- + ------ + -------- + -----------
               50        50        50           25

(C24) /* RESETTING OF OPTIONS */ ----all options(switches)are returned to
                                     their default settings.
      GLOBALSOLVE: FALSE$

(D25)                             BATCH DONE




(C26) "SOLUTION BY LAPLACE TRANSFORMS"$  ---only rational functions are
                                            considered here,but the strong
(C27) SUBST(Y(X),Y,DE);                     integration facility in MACSYMA
                                            would allow for more general
             2                              functions.
            D          D
(D27)       --- Y(X) - -- Y(X) - 6 Y(X) = SIN(X) ---(C27)causes the ODE to
              2        DX                             be displayed.
            DX

(C28) [ATVALUE(Y(X),X=0,1), ATVALUE('DIFF(Y(X),X),X=0,0)];---sets the boundary
                                                              conditions.Note the
(D28)                  [1, 0]                                 suppression of evalu-
                                                              tion at this stage.
```

```
(C29) LAPLACE(D29,X,S); ----this instruction causes the loading
                             of the corresponding disc.
LAPLAC FASL DSK MACSYM being loaded
loading done


        2
(D29) S  LAPLACE(Y(X), X, S) - S LAPLACE(Y(X), X, S)----the Laplace trans-
                                                         form of both sides
                                                         of the equn.is
                                                                  taken.
                                                        1
             - 6 LAPLACE(Y(X), X, S) - S + 1 = ------
                                                       2
                                                      S  + 1

(C30) LINSOLVE([%],['LAPLACE(Y(X),X,S)]);----the LT is determined
Solution                                         explicitly as a fn.of S.

                                         3    2
                                        S  - S  + S
(E30)      LAPLACE(Y(X), X, S) = --------------------  result is displayed;
                                     4    3      2
                                    S  - S  - 5 S  - S - 6

(D30)                               [E30] ----------- and stored.

(C31) ILT(E30,S,X); -------the inverse LT is taken.
                                             3 X           -2X
                7 SIN(X)    COS(X)    21 %E         14 %E
(D31)   Y(X) = - -------- + ------ + -------- + --------      final result.
                   50         50        50          25
```

(a)

```
procedure main
algebraic altran tdiff
algebraic(x:10, a:10, b:10, cos:10, sin:10, j0:10, j1:10, j2:10)
#  cos = cos(bx^2)
#  sin = sin(bx^2)
#  j0 = J_0(ax)
#  j1 = J_1(ax)
#  j2 = J_2(ax)
# indeterminates which depend on x
algebraic array(6) vars = (x, cos, sin, j0, j1, j2)
# their derivatives are
algebraic array(6) derivs =
  (1, -2*b*x*sin, 2*b*x*cos, -a*j1, a*j0 - j1/x, a*j1 - 2*j2/x )
# we also need
integer i
algebraic array(0:3) f
# now define the function and compute its derivatives
f(0) = j2 * cos
do i = 1,3
  f(i) = tdiff(f(i-1),vars,derivs)
doend
write f
end
```

The output of this program is

```
# f
( cos*j2 ,
- ( 2*x**2*b*sin*j2 - x*a*cos*j1 + 2*cos*j2 ) / ( x ) ,
- ( 4*x**4*b**2*cos*j2 + 4*x**3*a*b*sin*j1 -
x**2*a**2*cos*j0 - 6*x**2*b*sin*j2 +
3*x*a*cos*j1 - 6*cos*j2 ) / ( x**2 ) ,
( 8*x**6*b**3*sin*j2 - 12*x**5*a*b**2*cos*j1 -
6*x**4*a**2*b*sin*j0 + 12*x**4*b**2*cos*j2 -
x**3*a**3*cos*j1 + 12*x**3*a*b*sin*j1 -
3*x**2*a**2*cos*j0 - 24*x**2*b*sin*j2 +
12*x*a*cos*j1 - 24*cos*j2 ) / (x**3 ) )
```

(b)

```
procedure picard (f, x, y, y0, n)
integer i, n
algebraic x, y, f, y0, p = y0
do i = 1,n
   p = y0 + pint( f(y=p), x)
doend
return(p)
end
```

Then the following main program solves

$y'=ay$, $y(0)=1$

to tenth order

```
procedure main
algebraic (x:10, y:10, a:10) p
algebraic altran picard
p = picard( a*y, x, y, 1, 10)
write p
end
```

The result is the Taylor expansion of the solution $y=e^{ax}$, accurate to tenth order in x. The output is

```
# p
( x**10*a**10 + 10*x**9*a**9 + 90*x**8*a**8 +
720*x**7*a**7 + 5040*x**6*a**6 + 30240*x**5*a**5 +
151200*x**4*a**4 + 604800*x**3*a**3 +
1814400*x**2*a**2 + 3628800*x*a + 3628800 ) / 3628800
```

ALTRAN programs:(a)differentiation of $J_2(ax)\cos(bx^2)$;(b)Picard iteration($y'=f(x,y)$).

Some output from an ALTRAN program for Chebychev series solution of the hypergeometric equation.(From Geddes(1977)).

Output for Problem #11:

```
# DIFFEQ
       - ( X**2*DY(2) + X*DY(1)*MU(1) + X*DY(1)*MU(2) + X*DY(1) - X*DY(2) +
       Y*MU(1)*MU(2) - DY(1)*MU(3) )
# CONDN(1)
       YX(1) - 1
# CONDN(2)
       YX(2) + 1
# XSUB(1)
       0
# XSUB(2)
       1
# SETUP TIME IN SECONDS WAS
# TNEW
       2.60425
# HALFN
       2
# EQN
       - ( K**3*CK(-2) - 2*K**3*CK(-1) + 2*K**3*CK(0) - 2*K**3*CK(1) +
       K**3*CK(2) + K**2*CK(-2)*MU(1) + K**2*CK(-2)*MU(2) - 3*K**2*CK(-2) -
       2*K**2*CK(-1)*MU(3) + 4*K**2*CK(-1) + 2*K**2*CK(1)*MU(3) -
       4*K**2*CK(1) - K**2*CK(2)*MU(1) - K**2*CK(2)*MU(2) + 3*K**2*CK(2) +
       K*CK(-2)*MU(1)*MU(2) - K*CK(-2)*MU(1) - K*CK(-2)*MU(2) + 2*K*CK(-1) -
       2*K*CK(0)*MU(1)*MU(2) + 2*K*CK(0)*MU(1) + 2*K*CK(0)*MU(2) - 4*K*CK(0) +
       2*K*CK(1) + K*CK(2)*MU(1)*MU(2) - K*CK(2)*MU(1) - K*CK(2)*MU(2) +
       CK(-2)*MU(1)*MU(2) - 2*CK(-2)*MU(1) - 2*CK(-2)*MU(2) + 4*CK(-2) +
       2*CK(-1)*MU(3) - 4*CK(-1) - 2*CK(1)*MU(3) + 4*CK(1) - CK(2)*MU(1)*
       MU(2) + 2*CK(2)*MU(1) + 2*CK(2)*MU(2) - 4*CK(2) ) /
```

In the following, we shall compute the first few derivatives of the function

$$f(x) = J_2(ax) \cos (bx^2)$$

where J_2 is a Bessel function of the first kind. The informative is pretty short, and it can be used for computing the n first derivatives of any function composed of elementary functions and Bessel functions J_k with $k = 0, 1, 2, \ldots$ We only define and use one additional differentiational rule:

$$\frac{\partial}{\partial x} J_k(x) = \begin{cases} J_{k-1}(x) - \frac{k}{x} J_k(x) & \text{if } k \neq 0 \\ -J_1(x) & \text{if } k = 0 \end{cases}$$

(with a trivial generalization). We use the notation J(K,X) for $J_k(x)$.

```
"ПУСТ"ВD."ДЛ"I=1"Ш"1"ДО"N"ВЫП"
("ВЗЯ"∂/∂X(P3);L."ДИ";"ПРИМ"D.~,F;"ПРИВ"!!;"НА"L;
F."ВЫВ""ЗН""СТР",'D(',I,')=',P3)
"ГДЕ"D.(;E,K)∂/∂X(J(K,E))=("Е"K=0"ТО"-J(1,E)"ИНА"J(K-1,E)-K/E×J(K,E))
  ×∂/∂X(E)
"КОН"◊
```

Now we can easily compute the first five derivatives (we only give the beginning of the output):

```
"ВЫП""ВЗЯ"J(2,A×X)×COS(B×X↑2):N=5;"НА"ВD"КОН"◊
D(1)=J(1,A×X)×A×COS(B×X↑2)+(-2)×X↑(-1)×J(2,A×X)×COS(B×X↑2)+(-2)
×X×B×SIN(B×X↑2)×J(2,A×X)◊
D(2)=J(0,A×X)×A↑2×COS(B×X↑2)+(-3)×X↑(-1)×J(1,A×X)×A×COS(B×X↑2)+
(-4)×X×B×SIN(B×X↑2)×J(1,A×X)×A+6×X↑(-2)×J(2,A×X)×COS(B×X↑2)+6×B
×SIN(B×X↑2)×J(2,A×X)+(-4)×X↑2×B↑2×COS(B×X↑2)×J(2,A×X)◊
```

ANALITIK.(From Korpela(1976)).

```
"ПУСТЬ"PICARD.
"ДЛЯ"I=1"ШАГ"1"ДО"N"ВЫПОЛНИТЬ"
 ("ВЗЯТЬ"У0+∫(T=X0,X,F(T,W(T)));
  "ИНТЕГРИРОВАТЬ"-;
  "ПРИВЕСТИ"!!;
  "НАЗВАТЬ"W(X));
"ВЫВОД""СТРОКА",W(X)
"ГДЕ"
W(X)=У0
"КОНЕЦ"◇
```

In the following we solve
$y' = ay, \ y(0) = 1$
by ten iterations.

```
"ПУСТ"F(X,У)=A×У;X0=0;У0=1"КОН"◇
"ВЫП""ДРО""НЕ""ДЕ";N=10;"НА"PICARD"КОН"◇
W(X)=1+A×X+1/2×A↑2×X↑2+1/6×A↑3×X↑3+1/24×A↑4×X↑4+1/120×A↑5×X↑5·
1/720×A↑6×X↑6+1/5040×A↑7×X↑7+1/40320×A↑8×X↑8+1/362880×A↑9×X↑9+
1/3628800×A↑10×X↑10◇
```

ANALITIK.(From Korpela(1976)).

```
//list

'do'
'fractions''not''divided' ;
'take' 'Ux(I1xK0-I0xK1)' ;
'for' I=1 'step' 1 'until' 3 'do'
   ('take' d/dZ(Pz) ;
L.  'differentiate' ;
    'simplify' !! ;
    'apply' W.-,L2 ;
    'goto' L ;
L2. 'out' 35 , 'line', Pz)
'where'
U(Z) ;
I1(Z) ;
K0(Z) ;
I0(Z) ;
K1(Z) ;
W. d/dZ(I0) = I1x(AxZxU-A),
   d/dZ(K0) = (-(K1x(AxZxU+A))),
   d/dZ(I1) = I0x(AxZxU-A)+I1xU,
   d/dZ(K1) = (-(K0x(AxZxU+A)))-K1xU,
   d/dZ(U)  = (-(ZxU^3))
'end'
$

//new

//t int p

//comp
SYNTAX OK $

//


Pz=-(ZxU^3xI1xK0)+ZxU^3xI0xK1+2xI0xAxZxU^2xK0+I1xU^2xK0+(-2)x
K1xAxZxU^2xI1+K1xU^2xI0$



Pz=3xZ^2xU^5xI1xK0+(-6)xI0xAxZ^2xU^4xK0+(-3)xI1xU^4xK0xZ+6xK1x
AxZ^2xU^4xI1+(-3)xZ^2xU^5xI0xK1+(-3)xK1xU^4xI0xZ+4xI1xA^2xZ^2x
U^3xK0+(-4)xK1xA^2xZ^2xU^3xI0+(-4)xK1xAxU^2xI1$



Pz=3xZxU^5xI1xK0+(-15)xZ^3xU^7xI1xK0+30xI0xAxZ^3xU^6xK0+15xI1x
U^6xK0xZ^2+(-30)xK1xAxZ^3xU^6xI1+(-24)xI1xA^2xZ^3xU^5xK0+4xI1x
A^2xZ^2xU^4xK0+(-6)xZxU^4xK0xI0xA+24xK1xA^2xZ^3xU^5xI0+26xK1xA
xZxU^4xI1+(-3)xK0xU^4xI1+(-3)xZxU^5xI0xK1+15xZ^3xU^7xI0xK1+15x
K1xU^6xI0xZ^2+(-3)xI0xU^4xK1+8xI0xA^3xZ^3xU^4xK0+12xZxU^3xK0x
I1xA^2+(-8)xK1xA^3xZ^3xU^4xI1+4xK1xU^4xZ^2xI0xA^2+(-12)xZxU^3x
I0xK1xA^2+4xK0xA^2xU^2xI1+4xI0xA^2xU^2xK1$
```

A program example in 'English ANALITIK'.(From the Helsinki University of Technology).

```
A[3];G[15];H[15];C[63];E[255];F[255]
MODE=4

1:
TEXT:
Metric Identifier

Axially Symmetric Metric
------------------------


:
FOR I=0:1:15; G[I]=0; H[I]=0
REPEAT

A[0]=u[1]; A[1]=v[1]; A[2]=w[1]; A[3]=1
FOR I=0:1:3
     FOR J=0:1:I
          G[I+4J]=k[1]A[I]A[J]; G[4I+J]=G[I+4J]
     REPEAT
REPEAT
CLEAR(A[3])
G[0]=G[0]-f[1]
G[5]=G[5]-p[1]q[1]
G[10]=G[10]-p[1]r[1]

H[0]=-1/f[1]
H[3]=u[1]/f[1]
H[5]=-1/(p[1]/q[1])
H[7]=v[1]/f[1]
H[11]=-1/(p[1]/r[1])
H[12]=H[3]; H[13]=H[7]; H[14]=H[11]
H[15]=1/k[1] - (u[1]H[3]+v[1]H[7]+w[1]H[11])

FOR I=0:1:3
     FOR J=0:1:I
          ->2 IF G[I+4J]=0
          TEXT:G[:;PRINT(I);PRINT(J);TEXT:]:;PRINT(G[I+4J])
2:   REPEAT
REPEAT

FOR I=0:1:3
     FOR J=0:1:I
          K=I+4J
          E[K]=FSUB(FSUB(dG[K]/dt,0,q[t]),0,r[t])
          E[K+16]=dG[K]/dx
          E[K+32]=dG[K]/dy
          E[K+48]=0
          ->4 IF I=J
          L=J+4I
          E[L]=E[K]; E[L+16]=E[K+16]; E[L+32]=E[K+32]; E[L+48]=0
4:   REPEAT
REPEAT



FOR I=0:1:3
     FOR J=0:1:I
```

Part of a CAMAL program for relativity calculations.(J.P.Fitch).

(a)

```
CLEAR(W[3]);
W[0]=u;
W[1]=utan[v];                                          Several different sets of
W[2]=x;W[3]=arctan[y/x];                               co-ordinates are used in
->1 IF T=5;                                            succession,via loops.
TEXT:These are triangular co-ordinates:;
CLEAR(W[3]);
E=(u(a.2v.2+b.2).2+(1+v.2))/2uv.2;
F=u.2((a.2v.2-b.2).2+(1+v.2).2+2u(1+v.2)(a.2v.2+b.2)-2c.2v.2)/4u.2v.4;
W[0]=(E+F!(1/2))!(1/2);
W[1]=vW[0];
W[2]=(x.2+y.2-c.2)/(x.2-a.2)(y.2-b.2);
W[3]=y/x;
TEXT:This is one possible set of co-ordinates for studying stress:
problems for a rect.plate with a central circular hole:;
->1 IF T=6;
CLEAR(W[3]);U[1];
U[0]=0;U[1]=z;          A general TPS(possibly representing an approx.
FOR I=2:1:11;           conformal mapping fn.for a given domain)is used.
FOR J=2:1:I;            A formal reversion routine is applied to compute
U[0]=U[1]+a<J>z.J;      an approximate inverse to this mapping.
U[0]=SUB(SUB(U[0],x+iy,z),p<J>+iq<J>,a<J>);
PRINT[U[0]];REPEAT;
TEXT:U[0] represents,eq,an approx.conformal mapping function:;
FOR K=1:1:I-1;MAXORDER=K+1; this causes truncation of powers above
->2->;                      K+1.
2: Y=w;G=w.2;  This is a subroutine for approximate reversion.Note
FOR L=2:1:K+1; that subroutines cannot be called directly in CAMAL:
Y=Y+a<L>G;     they must be written into the program as input.
G=wG;
REPEAT;
REPEAT;
RETURN;
X=X+w-Y;
PRINT[X];
X=SUB(X,u+iv,w);
PRINT[X];
EXPAND(X,i,G[1]); This separates the real and imaginary parts of a fn..
PRINT[G[0]];PRINT[G[1]];
EXPAND(U[0],i,H[1]);
PRINT[H[0]];PRINT[H[1]];
```

(b)

```
e Laplacian is nowE[6] =      -2yxd<1,1>(xʟ2+yʟ2)ʟ(-(1/2)) /(xʟ2+yʟ2)
      +2yd<1,1>(xʟ2+yʟ2)ʟ(-(1/2)) /(yʟ2/(x)+x)
      +2d<1,0>(xʟ2+yʟ2)ʟ(-(1/2))
      +(xʟ2d<2,0>+yʟ2d<2,0>)(xʟ2+yʟ2)ʟ(-1)
      -(xʟ2d<1,0>+yʟ2d<1,0>)(xʟ2+yʟ2)ʟ(-(3/2))
      +2yxd<0,1>/(xʟ4+2yʟ2xʟ2+yʟ4)
      +d<0,2>/(xʟ2+yʟ2)
      -2yd<0,1>/(yʟ4/(x)+(xʟ3+2yʟ2x))

6] =  -d<1,0>(sin[v]ʟ2 +cos[v]ʟ2 )ʟ(-(3/2)) sin[v]ʟ2 /(u)  Again,SET instruction
      +d<2,0>(sin[v]ʟ2 +cos[v]ʟ2 )ʟ(-1) sin[v]ʟ2            should be imposed.
      -2d<1,1>(sin[v]ʟ2 +cos[v]ʟ2 )ʟ(-(1/2)) sin[v]cos[v]/(usin[v]ʟ2 +ucos[v]ʟ2
      -d<1,0>(sin[v]ʟ2 +cos[v]ʟ2 )ʟ(-(3/2)) cos[v]ʟ2 /(u)
      +d<2,0>(sin[v]ʟ2 +cos[v]ʟ2 )ʟ(-1) cos[v]ʟ2
      +2d<1,1>(sin[v]ʟ2 +cos[v]ʟ2 )ʟ(-(1/2)) cos[v]/(usin[v]+ucos[v]ʟ2 /(sin[v])
      +2d<1,0>(sin[v]ʟ2 +cos[v]ʟ2 )ʟ(-(1/2)) /(u)
      +2d<0,1>sin[v]cos[v]/(uʟ2sin[v]ʟ4 +2uʟ2sin[v]ʟ2 cos[v]ʟ2 +uʟ2cos[v]ʟ4 )
      -2d<0,1>cos[v]/(uʟ2sin[v]ʟ3 +2uʟ2sin[v]cos[v]ʟ2 +uʟ2cos[v]ʟ4 /(sin[v]))
      +d<0,2>/(uʟ2sin[v]ʟ2 +uʟ2cos[v]ʟ2 )

se are bipolarsW[0] =        asinh[v]/(-cos[u]+cosh[v])
```

(a)Part of a CAMAL program.(From Elvey(1978)).(b)A sample of CAMAL output.

Listing of deck LEGR

```
//LEGR   JOB   FORMAC,FORMAC,MSGLEVEL=1
//JOBLIB  DD   DSNAME=SYS1.SYSTEM.LINKLIB,DISP=OLD
//       EXEC  FORMAC
//SYSIN    DD  *
      LEGR: PROCEDURE OPTIONS(MAIN);
      FORMAC_OPTIONS;
      OPTSET(LINELENGTH=72);
      OPTSET(EXPND);
           PUT PAGE;

 /* GENERATE LEGENDRE POLYNOMIALS BY METHOD 1 */
           DO N=0 TO 10;
           LET(P("N")=DERIV((X**2-1)**"N",X,"N")/(2**"N"*FAC("N")) );
           END;

 /*GENERATE LEGENDRE POLYNOMIALS BY METHOD 2 */
           LET( Q(0)=1; Q(1)=X );
           DO N=2 TO 10;
           LET( N="N";
                Q(N)=(2*N-1)/N*X*Q(N-1) - (N-1)/N*Q(N-2)    );
           END;

 /*CHECK THAT P(N) = Q(N) AND PRINT OUT RESULTS */
           PUT LIST( '  LEGENDRE POLYNOMIALS' ); PUT SKIP(3);
           DO N=0 TO 10;
           LET( N="N" );
           IF IDENT(P(N);Q(N)) THEN PRINT_OUT(P(N));
                                ELSE STOP;
           END;
           END LEGR;
/*
```

Output from program LEGR

```
LEGENDRE POLYNOMIALS

P(0) = 1
--------
P(1) = X
--------
                2
P(2) = 3/2 X   - 1/2
--------------------
                        3
P(3) =  - 3/2 X + 5/2 X
------------------------
                 2         4
P(4) =  - 15/4 X   + 35/8 X   + 3/8
-----------------------------------
                        3          5
P(5) = 15/8 X - 35/4 X   + 63/8 X
-----------------------------------
                 2           4            6
P(6) = 105/16 X   - 315/16 X   + 231/16 X   - 5/16
--------------------------------------------------
                               3            5            7
P(7) =  - 35/16 X + 315/16 X   - 693/16 X   + 429/16 X
--------------------------------------------------------
                   2             4             6              8
P(8) =  - 315/32 X   + 3465/64 X   - 3003/32 X   + 6435/128 X   + 35/128
------------------------------------------------------------------------
                                3             5             7
P(9) = 315/128 X - 1155/32 X   + 9009/64 X   - 6435/32 X   + 12155/128 X
------------------------------------------------------------------------
9
--
                  2               4               6                8
P(10) = 3465/256 X   - 15015/128 X   + 45045/128 X   - 109395/256 X   +
------------------------------------------------------------------------
           10
46189/256 X    - 63/256
-----------------------
```

FORMAC.(From the 'FORMAC-73' Manual).

```
       CALL IDE: /* COMPUTE APPROXIMATE SOLUTION */
       PUT SKIP (2);
          DO I=1 TO N; PRINT_OUT(Y("I")); END;
          PUT SKIP(2); PUT LIST('TIME AT END'); PUT LIST(TIME);
       GO  TO START;
END: PUT LIST ('END OF JOB');
/*                                                    */
/* THE DIFFERENT VERSIONS OF IDE ARE INSERTED HERE */
/*                                                    */
IDE: PROC;
    DO I=1 TO N; LET(Y("I")=YO("I")); END;
DO KK=1 TO M;
    DO K=1 TO N; LET (K="K")   ;
            /* COMPUTE THE INTEGRAND */
        LET ( Y=EVAL(F(K),$X,X ));
        DO I=1 TO N; LET (Y=EVAL(Y,$Y("I"),Y("I"))  ); END;
            /* INTEGRATE */
        CALL IGRAL;
            /* YY IS THE INTEGRAL. ONLY KEEP THE NEW APPROXIMATION */
            /* AS FAR AS THE FIRST NEW TERM                          */
        LET(YY=(YO(K)+YY)-Y(K); L=LOWPOW(YY,X) );
        LET(Y(K)=Y(K)+COEFF(YY,X**L)*X**L );
        END;
IGRAL: PROC; /* THIS ROUTINE CALCULATES THE INTEGRAL OF THE  */
             /* POLYNOMIAL Y W.R.T. X BY REPLACING X**I BY   */
             /* X**(I+1)/(I+1), I=1....                      */
        LET (YY=EXPAND(X*Y); "L"=HIGHPOW(YY,X) );
        DO I=L TO 2 BY -1; LET (I="I";YY=REPLACE(YY,X**I,X**I/I));END;
END IGRAL;
END;
END IDE;
```

```
DY/DX=X**2+Y**2; Y(X=0)=0


           2            2
F(1) = $X    + $Y(1)
---------------------
YO(1) = 0
---------


                3            7              11                 15
Y(1) = 1/3 X    + 1/63 X     + 2/2079 X       + 13/219295 X       + 46/12442815 X
--------------------------------------------------------------------------------
19                       23                         27
    + 15178/66108676095 X      + 404/28332289755 X      + 190571/215183
--------------------------------------------------------------------------------
           31                             35
740539225 X      + 5854922/106515951641166375 X      + 1987447756/552
--------------------------------------------------------------------------------
                          39
4942451162729987125 X
----------------------
```

Part of a FORMAC program for iterative solution of differential equations;and a sample of output for a simple case.(FORMAC-73 Manual).

A Sample Program

The following shows an example of the interactive use of REDUCE on a PDP-10. Note that the asterisks in the first column are printed by the system to indicate that the program is ready for input and are NOT part of the REDUCE syntax.

```
REDUCE                                       load the program
REDUCE 2 (<system data>)...

*COMMENT A SAMPLE PROGRAM;                   comments are allowed
*X:=(Y+Z)**2;                                set X to (Y+Z)**2

      2             2                        here's the result printed
X := Y  + 2*Y*Z + Z                          because we used a semicolon
                                             as a terminator
*DF(X,Z,2);                                  differentiate X wrt Z twice

2                                            here's that result

*PROCEDURE FAC N;                            now define the factorial
*        BEGIN INTEGER M,N;                  function
*                M:=1;
*           A:   IF N=0 THEN RETURN M;
*                M:=M*N;
*                N:=N-1;
*                GO TO A
*         END;

*2**FAC 3;                                   we can omit the parentheses

64

*FAC (120);                                  or put them in with unary
                                             operators
<yes. big numbers do work!>

*SYMBOLIC;                                   enter symbolic mode

NIL                                          value returned in symbolic mode

*CAR ('(A));                                 compute the CAR of (A)

A                                            here's its value

*ALGEBRAIC;                                  return to algebraic mode

*X;                                          evaluate X again

 2             2
Y  + 2*Y*Z + Z                               it's still (Y+Z)**2

*END;                                        return to LISP
ENTERING LISP...                             so that you know
```

An illustrative program.(From the REDUCE-2 Manual).

```
OFF ECHO,MSG:TIME;
E(0)←- A*Y*(J1(Y)*COS(A*Y)+J21(Y)*SIN(A*Y))+D*SIN(A*Y)*J1(Y)
   +A*(J3(Y)*COS(A*Y)+J4(Y)*SIN(A*Y))-D*J2(Y)*COS(A*Y)-L↑2*H(Y):
LET DF(J1(Y),Y)=COS(A*Y)*H(Y),
    DF(J2(Y),Y)=SIN(A*Y)*H(Y),DF(J21(Y),Y)=DF(J2(Y),Y),
    DF(J3(Y),Y)=Y*COS(A*Y)*H(Y),
    DF(J4(Y),Y)=Y*SIN(A*Y)*H(Y);
FOR ALL X LET SIN(X)↑2=1-COS(X)↑2:
FOR I=1:4 DO E I←DF(E (I-1),Y):
E(5)←E(4)+A↑2*E(2)$E(6)←E(4)-A↑4*E(0)$E(7)←E(6)-2*E(5);
H(Y)←E↑(S*Y);FACTOR S:E(8)←E(7)/E↑(S*Y);
TIME;
```

Fig. 1. Differentiation program, with α translated to A, δ translated to D, λ translated to L.

```
OFF ECHO,MSG;

700 MS

                 2
E(0) :=  - (L *H(Y) + Y*A*J1(Y)*COS(Y*A) + Y*A*J21(Y)*SIN(Y*A) - A*

            J3(Y)*COS(Y*A) - A*J4(Y)*SIN(Y*A) - D*J1(Y)*SIN(Y*A) +

            D*J2(Y)*COS(Y*A))

           2  4           2  2                    2                    3
E(7) := L *A *H(Y) + 2*L *A *DF(H(Y),Y,2) + L *DF(H(Y),Y,4) - A *D*

                3
        H(Y) - A *H(Y) - A*D*DF(H(Y),Y,2) + A*DF(H(Y),Y,2)

         (Y*S)
H(Y) := E

         4  2    2       2               3   2
E(8) := S *L  + S *A*(2*L *A - D + 1) + A *(L *A - D - 1)

7350 MS
```

Fig. 2. Output from Fig. 1; E(7) is the differential equation and E(8) the characteristic equation.

From a REDUCE routine for solving integral equations(Loos(1971)).

Power Series

We obtain the solution of the differential equation

$$y'' + 2y'/x + y^n = 0$$

which is known as Emden's equation, using the formal power series package written for SCRATCHPAD by Arthur C. Norman. Here with, say, f declared to be a power series in x, Statements of the form 'f=E' cause the automatic definition of recurrence relations for successive coefficients f_i in $f=\Sigma_i f_i x^i$. The initial conditions at x=0 are y=1 and y'=0. The term y'/x involves a removable singularity at the origin.

SCRATCHPAD Conversation

"Declare x to be the independent variable"
```
psindvar=x

     PSINDVAR= X
```

"Declare f and y to be power series variables"
```
(f,y) ps

     F PS

     Y PS
```

"Emden's equation"
```
f=df<x;2>y+2*(df<x>y)/x+y**n

                         2( d Y)
               2            X         N
     (PS)  F=  d Y + --------- + Y
               X          X
```

"Ask for the first five terms of the power series for f=0 in x"
```
(f<i> for i in (0,...,4))

     2Y  MUST BE ZERO
       1
```

"So, let y_1=0"
```
y<1>=0

     Y = 0
      1
```

"Ask again for the first five terms"
```
(f<i> for i in (0,...,4))
```

```
                                   N                    N
                              N*Y   Y   + 20Y Y    N*Y   Y   + 30Y Y
            N                    0  2       4 0       0  3        5 0
    (1)  (Y    + 6Y  ,12Y  ,-------------------,--------------------, . . .)
            0      2     3           Y                    Y
                                      0                    0
```

"For each f_i to be 0, y_0 can evidently be any constant: choose $y_0=1$"
```
y<0>=1

     Y  = 1
      0
```

"Re-evaluate the first five terms"
```
(f<i> for i in (0,...,4))

     (2)  (6Y  + 1,12Y  ,N*Y  + 20Y  ,N*Y  + 30Y  , . . .)
             2         3    2       4    3       5
```

"Let y_i be the result of solving $f_{i-2}=0$ for y_i, $i \geq 2$"
```
y<i>=solve(f<i-2>,y<i>) , i in (2,3,...)

     Y  = SOLVE(F     ,Y ) WHEN i IN (2,3,...)
      i          i - 2  i
```

"Display the first 9 terms in the power series expansion of y"
```
(y<i> for i in (0,1,...,8))

                                       2               3        2
                           N     - 8N   + 5N      122N  - 183N  + 70N
     (3)  (1,0,-1/6,0,-----,0,-------------,0,----------------------)
                          120       15120               3265920
```

"Check by displaying the first 7 terms in the expansion for f"
```
(f<i> for i in (0,1,...,6))

     (4)  (0,0,0,0,0,0,0)
```

Generation of a Sylvester Matrix

The Sylvester matrix of two polynomials p and q of degrees n and m, respectively, in a 'main' variable x, is a square matrix of order m+n, formed using the coefficients of p and q. The determinant of the Sylvester matrix is called the resultant of p and q with respect to x. The resultant is of interest because of its connection with the greatest common divisor of p and q, and because of its use in finding common roots for p and q.

SCRATCHPAD Conversion

"Clear the environment"
```
)clear
```

"Define p"
```
p=sum<i=0;3>c<i>*x**i

         3
        ---     I
    P=  >    C X
        ---   I
        I=0
```

"Define q"
```
q=sum<i=0;2>d<i>*x**i

         2
        ---     I
    Q=  >    D X
        ---   I
        I=0
```

"n is the degree of p"
```
n=maxpower(x,p)

    N= MAXPOWER(X,P)
```

"m is the degree of q"
```
m=maxpower(x,q)

    M= MAXPOWER(X,Q)
```

"a_i is the coefficient of x^i in the polynomial p, for $i \leq n+1$, and 0, otherwise"
```
a<i> = if i <= (n+1) then covec(x,p) . (i-1) else 0

    A = COVEC(X,P) . (i - 1) WHEN i <= N + 1
     i
                          0 OTHERWISE
```

"b_i is the coefficient of x^i in the polynomial q, for $i \leq m+1$, and 0, otherwise"

```
b<i> = if i <= (m+1) then covec(x,q) . (i-1) else 0

     B = COVEC(X,Q) . (i - 1) WHEN i <= M + 1
      i
                        0 OTHERWISE
```

"$a=(c_0,c_1,\ldots,c_m,0,\ldots,0)$"

```
a=(a<i> for i in (1,2,...,m+n))

     A= (A  FOR I IN (1,2,...,M + N))
          I
```

"$b=(d_0,d_1,\ldots,d_n,0,\ldots,0)$"

```
b=(b<i> for i in (1,2,...,m+n))

     B= (B  FOR I IN (1,2,...,M + N))
          I
```

"Form the m by m+n matrix z, each row consisting of the vector a shifted one further place to the right"

```
z=(rotate(1-i,a) for i in (1,2,...,m))

     Z= (ROTATE(1 - I,A) FOR I IN (1,2,...,M))
```

"Form the n by m+n matrix y, each row consisting of the vector b shifted one further place to the right"

```
y=(rotate(1-i,b) for i in (1,2,...,n))

     Y= (ROTATE(1 - I,B) FOR I IN (1,2,...,N))
```

"Form the Sylvester matrix consisting of the catenation of the rows of z with those of y"

```
catenate<1>(z,y)

              *C  C  C  C  0 *
              * 0  1  2  3   *
              *              *
              *0  C  C  C  C *
              *    0  1  2  3*
              *              *
       (1)    *D  D  D  0  0 *
              * 0  1  2      *
              *              *
              *0  D  D  D  0 *
              *    0  1  2   *
              *              *
              *0  0  D  D  D *
              *       0  1  2*
```

Example 2: The Kepler Equation

The iterative solution of the Kepler equation

$$E = u + e \cdot \sin(E)$$

is a simple but nontrivial formula manipulation task, and is therefore often cited as an example. With E = u + A, the iteration based on the equation in the form

$$A = e \cdot \sin(u+A)$$

requires the expansion of sin(u+A) into a series. In the program below, this is accomplished with the standard function "Taylor". At the same time "fmod[1] := 1" forces the linearisation of the sin/cos terms and "smod[4] := i+1" guarantees appropriate truncation.

```
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 .          KEPLER EOUATION EE = U + E*SIN(EE)                             .
 .          -----------------------------------                            .
 .'BEGIN'                                                                  .
 .    A := {0: 0, 10:};                                                    .
 -                                              (FMOD[1] := 1);            .
 .    'FOR' I := 0 : 7 'DO' 'BEGIN'                                        .
 .                                              (SMOD[4] := I + 1);        .
 .        A[I+1] := TAYLOR(E*SIN(U+X), X, I, A[I]);                        .
 .    'END';                                                               .
1+'END';                                                                   .
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
                                                     START EXECUTION
A       :=  {0: 0, 10:};

A[1]    :=  E*SIN(U);

A[2]    :=  E*SIN(U) + 1/2*E^2*SIN(2*U);

A[3]    :=  E*SIN(U) + 1/2*E^2*SIN(2*U) + 3/8*E^3*SIN(3*U) - 1/8*E^3*SIN(U);

A[4]    :=  E*SIN(U) + 1/2*E^2*SIN(2*U) + 3/8*E^3*SIN(3*U) - 1/8*E^3*SIN(U)
           + 1/3*E^4*SIN(4*U) - 1/6*E^4*SIN(2*U);

A[5]    :=  E*SIN(U) + 1/2*E^2*SIN(2*U) + 3/8*E^3*SIN(3*U) - 1/8*E^3*SIN(U)
           + 1/3*E^4*SIN(4*U) - 1/6*E^4*SIN(2*U) + 125/384*E^5*SIN(5*U)
           - 27/128*E^5*SIN(3*U) + 1/192*E^5*SIN(U);

A[6]    :=  E*SIN(U) + 1/2*E^2*SIN(2*U) + 3/8*E^3*SIN(3*U) - 1/8*E^3*SIN(U)
           + 1/3*E^4*SIN(4*U) - 1/6*E^4*SIN(2*U) + 125/384*E^5*SIN(5*U)
           - 27/128*E^5*SIN(3*U) + 1/192*E^5*SIN(U) + 27/80*E^6*SIN(6*U)
           - 4/15*E^6*SIN(4*U) + 1/48*E^6*SIN(2*U);

A[7]    :=  E*SIN(U) + 1/2*E^2*SIN(2*U) + 3/8*E^3*SIN(3*U) - 1/8*E^3*SIN(U)
           + 1/3*E^4*SIN(4*U) - 1/6*E^4*SIN(2*U) + 125/384*E^5*SIN(5*U)
           - 27/128*E^5*SIN(3*U) + 1/192*E^5*SIN(U) + 27/80*E^6*SIN(6*U)
           - 4/15*E^6*SIN(4*U) + 1/48*E^6*SIN(2*U) + 16807/46080*E^7*SIN(7*U)
           - 3125/9216*E^7*SIN(5*U) + 243/5120*E^7*SIN(3*U) - 1/9216*E^7*SIN(U);

A[8]    :=  E*SIN(U) + 1/2*E^2*SIN(2*U) + 3/8*E^3*SIN(3*U) - 1/8*E^3*SIN(U)
           + 1/3*E^4*SIN(4*U) - 1/6*E^4*SIN(2*U) + 125/384*E^5*SIN(5*U)
           - 27/128*E^5*SIN(3*U) + 1/192*E^5*SIN(U) + 27/80*E^6*SIN(6*U)
           - 4/15*E^6*SIN(4*U) + 1/48*E^6*SIN(2*U) + 16807/46080*E^7*SIN(7*U)
           - 3125/9216*E^7*SIN(5*U) + 243/5120*E^7*SIN(3*U) - 1/9216*E^7*SIN(U)
           + 128/315*E^8*SIN(8*U) - 243/560*E^8*SIN(6*U) + 4/45*E^8*SIN(4*U)
           - 1/720*E^8*SIN(2*U);
```

SYMBAL program, with sample output. (From Engeli(1975)).

This problem and its solutions are described in the SIGSAM Bulletins No. 32,33 and 34. The first solution shown here with the complete listing uses straightforward Taylor series expansions for exp(t), exp(-t), sqrt(1+t), 1/(1+t), (1+t)↑n leaving only multiplications and additions of series to do the rest. The program is quite straightforward but not particularly fast.

In a second version of the program, the series are represented as lists of coefficients. All of the operations, including additions and multiplications, are programmed on a power by power basis using available algorithms (e.g. Knuth vol. II). Only the program-listing is given for this version.

The second version is faster by a factor of three as may be seen from table II below.

	Taylor series version		Series in list form	
m	Time for Computation	Time for Printing	Time for Computation	Time for Printing
3	1.703	0.043	0.791	0.062
4	4.345	0.202	1.777	0.247
5	11.169	0.469	3.885	0.499
6	-	-	9.197	1.013

Table II: Times for computation of SIGSAM Problem No. 8 on CDC 6500 using two program versions

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
.        SIGSAM PROBLEM NO 8    (TAYLOR)
.        ------------------------------
.'BEGIN'
.    'NEW' T, U, N, Q, T1, T2, T3, HN;
-                                             (PMOD[1] := 0);
.    M := 5;
.                                             SMOD[4] := M;
.    T1 := TAYLOR(EXP(T), T, M);
.    T2 := TAYLOR(EXP(-T), T, M);
1+   B := (1 - Q + Q*U)*T1 + (1 - Q - Q*U)*T2;
.    R := 2*Q*TAYLOR(SQRT((1+T)), T, M, (B^2 - 4*(1 - 2*Q))/(4*Q^2) - 1);
.    X := (B + R)/2;
.                                             SMOD[4] := M-2;
.    H := DELAYED(((1 + U)*T1 + (1 - U)*T2 - B + R)/2);
-    D := H/(2*Q)*TAYLOR(1/(1+T), T, M-2, DELAYED(R)/(2*Q) - 1);
.                                             SMOD[4] := M;
.    E := TAYLOR((1+T)^N, T, M, X-1);
.    T3 := TAYLOR(EXP(T), T, M, -N*U*T);
.    E := E*T3 - 1;
2+   HN := COEFF(D*E, T);
.                                             SMOD[1] := 1;
.                                             SMOD[2] := 10;
.                                             (PMOD[1] := 1);
.    'FOR' I := 2 : M 'DO' HN[I] := DELAYED(HN[I]);
-'END';
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                                                   START EXECUTION       0. 0.894
HN[2]  :=  -1/2*(U^2*N - U^2*N*Q - N + N*Q)/Q;
                                                                         0.11.912
HN[3]  :=  -1/6*(3*U*N - 6*U*N*Q + 2*U*N*Q^2 - 3*U^3*N + 6*U^3*N*Q - 2*U^3*N*Q^2)/Q^2;
                                                                         0.11.969
HN[4]  :=  1/24*(24*U^2*N - 72*U^2*N*Q + 56*U^2*N*Q^2 - 8*U^2*N*Q^3 - 6*U^2*N^2*Q
          + 12*U^2*N^2*Q^2 - 6*U^2*N^2*Q^3 - 18*U^4*N + 54*U^4*N*Q - 42*U^4*N*Q^2
          + 6*U^4*N*Q^3 + 3*U^4*N^2*Q - 6*U^4*N^2*Q^2 + 3*U^4*N^2*Q^3 - 6*N + 18*N*Q
          - 14*N*Q^2 + 2*N*Q^3 + 3*N^2*Q - 6*N^2*Q^2 + 3*N^2*Q^3)/Q^3;
                                                                         0.12.143
HN[5]  :=  -1/60*(135*U*N - 230*U*N*Q + 100*U*N*Q^2 - 8*U*N*Q^3 + 15*U*N^2 - 45*U*N^2*Q
          + 40*U*N^2*Q^2 - 10*U*N^2*Q^3 - 360*U^3*N + 610*U^3*N*Q - 260*U^3*N*Q^2
          + 20*U^3*N*Q^3 - 30*U^3*N^2 + 90*U^3*N^2*Q - 80*U^3*N^2*Q^2 + 20*U^3*N^2*Q^3
          + 225*U^5*N - 380*U^5*N*Q + 160*U^5*N*Q^2 - 12*U^5*N*Q^3 + 15*U^5*N^2
          - 45*U^5*N^2*Q + 40*U^5*N^2*Q^2 - 10*U^5*N^2*Q^3)/Q^3;
                                                                         0.12.354
```

SYMBAL program with sample output.(From Engeli(1975)).

14. Symbolic analysis(the mathematical use of symbolic computation).

The object of this section is to offer some suggestions for the use of symbolic computation in an intrinsically mathematical context,as compared with the usual approach which,though undoubtedly of great importance in many calculations(involving perturbation,iteration,reversion,etc.),does not make anything like full use of the very powerful facilities now available.Symbolic computation is inherently mathematical as a form of computing;but,for the most part,it has not been developed by mathematicians.This is reflected in the visions of what is 'desirable',both on the level of internal implementation,and in projected applications.Indeed,the dominant design criteria are concerned with economy of storage,flexibility of editing facilities,de-bugging,and elegance of data-handling.All of these facets of manipulation are,of course,basic for the mathematical operations,but often they are treated as ends in themselves.Even when mathematical matters are considered,the greatest effort has been devoted to dealing effectively with 'largescale problems of standard type'(such as the solution of systems of linear algebraic equations,or of ordinary differential equations with constant coefficients).An inspection of the lists of 'library routines' confirms this(understandable)preoccupation with standard techniques. One should add,however, that MACSYMA offers an increasingly wide range of basic,built-in facilities,while REDUCE,which offers relatively few fixed routines,has,nevertheless,an impressive collection of procedures contributed by users,and potentially available(provided that the corresponding programs have been distributed generally).Other systems fall somewhere between these two positions.In spite of this,the'analytical use of packages',in the sense to be outlined in this section,has hardly been exploited at all—even implicitly(but see the papers by Stoutemyer listed in the bibliography,for a step in this direction; and the agenda for the next 'SEAS/SMC' meeting,to be held in Antwerp,January,1981, where a further trend towards 'controlled abstraction' is noticeable).Again,there are very few analytically nontrivial algorithms in regular use(only those for factorization,GCD and formal integration being of appreciable sophistication,complexity and depth--see Section 12).

The great progress made in the design of hardware(and of the most basic algorithms,on which all of the more elaborate constructions rely)suggests that a new phase in symbolic computation should be inaugurated.The computational activity in this new phase may be called,aptly,symbolic analysis(which covers techniques customarily associated with 'Alegebra','Geometry' and 'Analysis',as well as other fields).The term 'inferential analysis'was used by Wang(1960)in connection with 'mechanical theorem-proving',the idea being to derive theorems of the propositional calculus,and other deductive systems,using general procedures for forming 'conclusions' from 'premises'--and to identify cases where such deduction cannot be made.

In this way,Wang was able to produce proofs for most of the theorems in the first five chapters of Principia Mathematica(Russell and Whitehead(1910)).His aim was to extend these methods to more sophisticated domains such as number theory,or even calculus.This approach to 'symbolic mathematics' is very different from that to be developed here,since there is no interaction of programmer and machine.The 'axioms' and other basic rules are 'given' to the machine—after which,it sifts the propositions it is offered into 'theorems' and 'nontheorems'.The pioneering efforts of Wang(1960),Burks et al.(1954),Collins(1957),Davis(1957)and Newell et al.(1957)were stimulated by the advent of fast calculating machines;but work on automatic theorem-proving still continues(see,e.g.,volumes of the IBM Journal for Research and Development,and of the Journal of Symbolic Logic).

14.2. Although 'mechanical theorem-proving' and 'symbolic analysis' are very different,certain ideas raised briefly by Wang,for general consideration,do have some relevance here.These ideas include:scope for investigation(as determined by a combination of 'capacity for abstract analysis',and 'capability of handling large expressions');centrality of a concept or theorem(as typified,e.g., by its 'frequent occurrence in proofs'or its 'range of application'—more formally, for a concept(resp.,theorem),by its existence as a short expression(resp.,statement),for which all known equivalent expressions(resp.,proofs),are 'far longer'); and,notions of approximate proof (for which no candidates are suggested by Wang, though he claims that various formalizations could be given).Although it seems to me most unlikely that nontrivial,yet logically rigorous modes of approximate proof can be found,the idea of various'stages of (in)completeness' of proofs,and of other intuitively appealing concepts,is of relevance in symbolic analysis as I envisage it—indeed,such 'exploratory activities' play an essential,though subsidiary, part in it;so, a few tentative remarks in this area are made in Section 14.3.The question of scope is very important in symbolic computation.It is plausible that the development of mathematica techniques has been conditioned strongly by man's limited capability of handling large expressions.The tendency is,almost always, condensing information into 'tractable forms'.Of course,in many instances this is very desirable;but not invariably.The most striking counter-example is the 'computer-assisted proof' of the Four Colour Conjecture(Appel and Haken(1977a,b);but it is probable that many more problems which have defeated all conventional attacks will succumb,eventually,to this partnership of powerful abstraction backed by comparatively unlimited computational capacity.Moreover,this consideration applies just as well to the use of routine tools of Algebra and Analysis as to the testing of 'exotic methods'.For example,in amny contexts,the mere application of some basic inequality is ruled out by the size of the expressions involved;yet,after due simplification,the result may well suggest a further analytical strategy quite unmotivated by the forms of the original expressions.Thus,one aim of symbolic analysis is to increase the scope of investigation associated with abstract procedures.

The notion of centrality seems to have a high subjective content— logical' definitions' notwithstanding;but its possible uses in symbolic analysis serve principally as a guide to the selection of results that are 'especially relevant' for the problem under investigation,so precise definition is not essential.This is discussed further in Section 14.4.

<u>14.3.</u> Plainly,no unimpeachable definitions of 'approximate proof' are to be expected,since there is,in many contexts,substantial disagreement over what costitues an 'absolute proof'.Nevertheless,in relation to dialogue investigation involving mathematician(s) and machine(s),several variants of this idea could be valuable.On this understanding,the following suggestions for types of quasi-proof are offered—one aim being to stimulate(or goad)readers to produce more satisfactory definitions.(All of the following schemes for investigation—and many more—are covered by the vague term,'heuristics';but the use of 'quasi-proof' seems to have the right connotation for the present work).

<u>14.3.1. Quasi-proofs based on the use of'probability logic'</u>.
It is possible to construct systems of logic in which truth values may be assigned arbitrarily within the interval $[0,1]$,subject to natural consistency requirements. (See,e.g.,the book by Rescher(1969)). It may be shown that countably-valued and finitely-valued logics are included in this scheme;they are obtained by introducing suitable step functions.The usefulness of many-valued logics has been called in question more than once(see e.g.,Scott(1976),who proposes an interpretation in terms of 'degrees of error').Since mathematical theorems are tautologies,there is no question of assigning fractional truth values to the proofs of theorems.For,if such a theorem,say '$p \Rightarrow q$'constitues a valid inference of p from q,then this validity is independent of the truth value(s) of the(components of the)premise,p.However,even though the theorem has this tautological form,the 'respect' accorded to its conclusion,q,as a 'known property of specified mathematical objects' <u>does</u> depend on the status of its premise(s),which may contain conditions whose validity is unknown—even,suspect.It is for such 'theorems' as this that it could make sense to assign a truth value,say ν,other than 0 or 1,to(the collection of premises) p—in which case,since the inference itself is tautological,the conclusion,q,'inherits the truth value,ν,from p'. If p comprises n propositions,for each of which the truth value is known to be $\geqslant 1-\varepsilon$,then it may be shown(see Suppes--p.54 of Hintikka and Suppes(1966))thatthe truth value of q,the conclusion,is $\geqslant 1-n\varepsilon$;thus,the truth value,ν,need not be(and generally,is not)known precisely;and,indeed,the systematic estimation of such 'compound truth values' is the major problem in a full development of this scheme.Notice that,in this simplistic approach,truth values and probabilities are used almost interchangeably,no distinction being made between merely assigning probabilities to statements,on the one hand,and calculating their truth values within a system of 'probability logic,on the other.However,there are

many subtle(and contentious)arguments involved.Several attempts have been made to analyse the properties of 'valid inference',usually,on a probabilistic basis.Some of these ideas are considered in Hintikka and Suppes(1966).Apart from the bounds on the validity of statements inferred from imperfectly known(compound)premises--as above—they include:measures of 'strength of evidence';consistent assignment of probabilities to arbitrary logical formulae(within a given probability system);the logic of conditionals;and,notions of 'reasonable consequence'(an inference being 'acceptable',if it is impossible for its premises to have 'high' probability, while its conclusion has 'low' probability.Yet another approach,this time to the problem of assigning probabilities directly to mathematical statements,has been made by Jeffreys(1957,1961),who uses a weighting procedure to rank differential equations by their 'complexities'(defined as the sum of the order,degree and all of the moduli of the coefficients).This idea is used both in the calculation of probabilities,and in an attempt to justify the basic laws of physics by the relatively low complexities of the corresponding differential equations.Although Jeffreys' scheme is incomplete and speculative,it is most suggestive,and raises fascinating questions,some of which are of general mathematical interest.

From these arguments about probabilistic logic(and fractional truth values),it follows that a'conclusion' can have any truth value in [0,1],depending on the truth values of(the components of)its premise(s)—assuming the deduction itself to be valid;and this seems eminently reasonable,even if it may not be convenient! One obvious example of this kind of situation is given by the proofs of various statements,q,in analytic number theory,'on the(generalised) Riemann Hypothesis. Another example(but of a different kind)concerns statements involving the '(generalised) Continuum Hypothesis'(though this was shown,by Cohen(1963),to be independent of the axioms of set theory,as usually formulated).However,it is clear that countless instances are met even in routine investigations,where conclusions about objects of principal interest depend on unproven hypotheses(subsidiary conditions involving 'parameters'--which may be numbers,functions,matrices,...).If a complicated proof requires the use of several of these 'condition-dependent theorems', then it is desirable to be able to estimate the validity of ρ,the 'final result'.If this estimate is favourable,then further analysis(using ρ as a condition-dependent hypothesis)may be worthwhile.Ultimately,if some conclusion of sufficient(practical or theoretical)interest is reached,with 'high validity', then attempts to find a rigorous proof may be made.<u>In short</u>:this approach should be seen as <u>an aid to the discovery of interesting theorems</u>.As such,it is well suited for use in symbolic analysis,provided that an adequate level of consistency can be maintained;and this can be ascertained only'by experiment'.

14.3.2. Quasi-proofs using occasional probabilistic estimates.

This mode of quasi-proof is exemplified by the Appel/Haken investigation of the Four Colour Conjecture(1977).Recent remarks by Haken(1978)on the isolated use of 'averaging procedures',combined with rigorous arguments,amount to interpreting ignorance of the value of some entity by its being(uniformly)randomly distributed over an appropriate domain of 'possible values'.Other forms of (nonuniform)distribution may be indicated when extra information is available(but,insufficient for the precise determination of the object in question).As Haken says,one has the alternative of either making no prediction at all(about a property),or else,of making some 'plausible assumption',on the basis of which qualified predictions may be made.In some cases,fairly straightforward probabilistic estimates yield striking results(for instance,Haken's estimate,based on properties of the sequence of integer Kth powers,that Fermat's Last Theorem has probability $< 3^{-N}$ of being false—where $N = 125001$ is the least integer for which the 'Theorem' has not been proven yet).A less crude probabilistic analysis bearing on Fermat's Last Theorem was given by Erdos and Ulam(1971).Their arguments suggest that,even if the result is true,there are,in general,only finitely many solutions(which does not seem unreasonable,in view of Baker's remarkable bounds on the number of solutions of certain Diophantine equations in two variables:see Baker(1974)).For other celbrated problems(e.g.,the Riemann Hypothesis),no simple,but credible,estimates seem to be obtainable.Of course,all results of this type depend on various 'randomness assumptions',and,to this extent,they lack logical force;though they are of great value for deciding whether it is whorthwhile trying to prove a result rigorously(or, to find a counter-example),either by further analysis,or else,by direct computation.

A somewhat different use of probabilistic assumptions is embodied in the concept of'statistical metric spaces',introduced by Menger,modified by Wald,and developed by Scweitzer and Sklar(1960),among others(they give references,and a brief history of the subject).A recent contribution to this theory is by Morrel and Nagata(1978).The basic idea is to replace the notion of a single metric(or distance function)by a probability distribution,F_{AB},where A and B denote points,and, for $x \geqslant 0$, $F_{AB}(x) := \text{Prob.}\{\text{distance}(A,B) < x\}$.This leaves the concept of 'exact distance' undefined;but it does assign a probability to the statement:'distance(A,B) lies between $x-\delta$ and $x+\delta$,for any(small)positive number,δ.'There are,undoubtedly,circumstances in which the concept of a 'point' cannot be well-defined;and others,where 'exact measurement' has little meaning.It is in such cases that calculations performed within the framework of statistical metric spaces could be valuable in constructing quasi-proofs,in the sense of this subsection.

Somewhat in the same mould(though not using statistical ideas directly)is the large literature on so-called fuzzy topological spaces,and on fuzzy subsystems (based on an idea of Zadeh(1965),which has given rise to a great variety of abstractions and applications--see,e.g.,papers in the Journal of Mathematical Analysis and Applications,and in Information and Control).For the most part,the abstractions are concerned with recreating parts of general topology,measure theory and probability theory,with fuzzy objects replacing the usual ones.Corresponding notions of convergence,compactness,continuity,etc.,are defined,and their special properties are studied.The original idea was formulated in the context of electrical engineering;but a simple mathematical definition is as follows(see,e.g., Hutton(1977)):let $L^* := (L, \leqslant, ')$ denote a completely distributive lattice,with order-reversing involution, $'$ .Then,an L^*-fuzzy set on a set,X,is any map,$A:X \to L$, where L is interpreted as a set of truth values and,for each x in X,$A(x)$ is taken as the degree of membership of x in the fuzzy set,A.(Thus,when $L = \{0,1\}$,the collection of fuzzy sets corresponds to the characteristic functions of ordinary sets). Once'union','intersection' and 'complement' have been defined for fuzzy sets(using the basic operations in L^*),fuzzy topological spaces are defined in the obvious way--a specified collection of 'fuzzy-open sets' being called a (fuzzy) topology for the underlying set.Two recent papers,giving references to some of the other ramifications of fuzziness,are Hutton and Reilly(1980),and Lowen(1979).A journal, Fuzzy Sets and Systems,has been published by North-Holland since 1978.For the purposes of this paper,it may be possible to formulate some of the concepts of quasi-proof precisely in terms of fuzzy sets.Although this will not achieve any direct results,it will pinpoint the'zones of incompleteness'in a systematic way,so that the logical experiments I have advocated can be performed with maximum safety!(It is worth remarking that many 'applications' of fuzzy concepts to more or less practical problems seem neither necessary,nor convincing:most of the nontrivial ones could be handled without introducing fuzziness at all.However,the purely mathematical developments,though of doubtful interest in some cases,are dealt with rigorously).

Quite different in origin is the circle of ideas and methods which are studied under the collective name,interval mathematics.This subject first emerged as a separate area of research as a result of the fundamentalwork of Moore(1968).Recent work may be found in the Karlsruhe symposium(Nickel(1975)),and in another book by Moore(1979).The basic ideas emanate from attempts to formalise error-arithmetic in digital computing.Among other things,this entails defining a metric topology of intervals,and using intervals as the basic entities in matrix computations,and in iterative methods for the solution of nonlinear equations.The theory has reached a high level of sophistication,and,even though its emphasis is on numerical work, it is certain to play a useful rôle in symbolic computation,where various forms of indeterminacy are present.

14.3.3. Quasi-proofs based on simplifying assumptions.

This mode of quasi-proof is very common in some parts of theoretical physics,where precise information is unobtainable,and exact structural relationships are either unknown,or else prohibitively complicated,computationally.The relevance of this kind of 'proof'within mathematics is comparable with that of arguments in which (somehow)all premises are assigned probabilities.However,the idea is quite distinct, since,once the assumptions have been made,all deductions are rigorously correct— and,most importantly,no attempt is made to assign any probability to the(correctness of the)assumptions adopted.(An excellent example is furnished by the 'proof' of the 'Prime Number Theorem' due to Courant and Robbins(1969),where it is assumed that there is a smooth density function governing the distribution of primes--all subsequent deductions being totally respectable.It is interesting to note that the 'density assumption' was suggested by Gustav Hertz,an experimental physicist.Moreover,almost every 'model' used in theoretical physics incorporates some assumption designed purely to render it mathematically tractable--ranging from crude 'replacement' to highly sophisticated approximation schemes).In spite of the obvious shortcomings of this approach,it should be used more often in the preliminary stages of mathematical investigations.The use of symbolic analysis would allow a variety of assumptions to be tested in an exploratory manner,at comparatively little cost(in time and money);and this could well give valuable indications for an eventual rigorous analysis.

14.3.4. Quasi-proofs with acknowledged gaps.

The idea here is to consider 'sketches of proofs',in which gaps are closed successively,until a complete,rigorous proof is attained.This situation(but with many unacknowledged gaps)is encountered frequently in scientific applications,where 'common sense'(that most unreliable of guides)dictates that certain results are 'obviously true',and the so-called proofs try to formalise this conviction.For all the,deserved,disreputability of this strategy,it does contain a germ of usefulness in relation to symbolic analysis.For instance,a sketch-proof may be structurally sound,even if it has several gaps.These gaps constitute results which must be obtained to convert the sketch into a full proof.There is no reason why such controlled speculation should not be valuable if used with caution.Of course,if more direct methods are available—and tractable--then they are to be preferred, but it is highly desirable to have some lower-grade procedure in reserve.

14.3.5. Rigorous proofs of weak forms of propositions.

This idea has many variants. The aim, for a given proposition, p, is to formulate (and then prove rigorously), propositions, say, $\tilde{p}$, which are, in a specified sense, 'close to p'. In favourable cases, p itself may be derived from a sequence, $\{\tilde{p}_k\}$, of weaker propositions, where each $\tilde{p}_k$ is relatively simple to prove. At worst, this should yield arbitrarily good 'approximations to p'; and, when only a finite number of $\tilde{p}_k$ are required, the exact form of p can be established. This strategy is used in factorization algorithms of the 'modular' or 'Hensel' types: see Section 12. Examples of the use of weak forms of 'classical procedures' abound in mathematics; much of functional analysis may be viewed in this light. A few obvious, but seminal examples include: the replacement of classical differentiability by weak forms (leading to the theory of generalized functions, Sobolev spaces, etc.); replacement of the Riemann integral by the Lebesgue integral (allowing a rigorous development of the theory of stochastic processes, among many consequences); and, the introduction of various nonclassical topologies, in terms of which, continuity, compactness, etc., can be formulated in contexts far more varied than the original definitions permitted. The relevance of all this to symbolic analysis is that it offers models for possible modes of reasoning which may be used to attack problems otherwise inaccessible. One more variation on this theme amounts to modifyin the domain over which a problem is defined, and then solving the problem fully--an excellent example being the proof by Weil(19) of the Riemann Hypothesis for finite fields, though this has not so far contributed directly to solving the problem for the general, complex number field (the best 'quantitative result' is due to Levinson(1974)). For a systematic use of 'p-adic methods' in number theory, see, e.g., Borevich and Shafarevich (1966). There is wide scope in symbolic analysis for experimental exploration of weak forms of propositions, and their relation to stronger forms.

None of these notions of quasi-proof is claimed to be more than suggestive--either of conjectures offering topics for fruitful investigation; or else, of possible methods of attack on problems whose importance is known already from earlier, rigorous results. In spite of this, all of these concepts of quasi-proof must be analysed and formulated with precision before they can be used with confidence, even as peripheral aids in large-scale studies. The only imprecision that is tolerable in this context is that contained in the randomness, simplifying or weakening assumptions injected deliberately. All subsequent must meet the highest standards of rigour. Thus, although there would appear to be little hope of constructing a rigorous calculus of 'partial implication', with direct application to mathematical proof, considerations such as those outlined here could be of value as aids to the ultimate, rigorous proofs of results that, probably, would never be identified otherwise.

14.4. Theorems as 'operators'.

Any mathematical theorem may be regarded as a 'processor' of its premises—the 'output' being its conclusions.Taken in isolation,this observation is trivial;it is made here because this view of inference fits well with the operational scheme of symbolic analysis,as I envisage it.In these terms,procedures 'act on' expressions(such as vectors,matrices,tensors,groups,anlaytic functions—and,in general, on arbitrary,well-formed expressions within the underlying logical system);whereas, theorems 'act on' statements(or logical formulae,such as,'let the condition,C,hold). As remarked in Section 14.3,the truth value of '$p \Rightarrow q$' is independent of that of p;and this is unsatisfactory,especially in the context of hypothesis-formation in scientific research(including applied mathematics).Even in 'pure mathematics', there is room for the use of nonstandard logical frameworks(see,e.g.,Davis(1977); also,Luxemburg(1969,1972,1973)),and this may be combined,in symbolic analysis, with the application of standard theorems and techniques.The successive application of theorems corresponds to composition of operators,and converse theorems,to inverse operators—an analogy which can be helpful in describing complicated procedures.

My main proposal is to extend the 'environment of evaluation' to an 'environment of investigation'.Apart from the collection of simplification rules and side relations usually imposed,carefully chosen theorems,deemed especially relevant to the problem in hand,should be incorporated(as data),for optional use on current (sub)-expressions,at any stage of a computation.It appears that SCRATCHPAD,with its deliberately conversational mode of use(see Section 9),is structurally suitable for this approach;indeed,its design seems to have been motivated partially by such aims.However,for my purposes,it is a matter of employing this strategy widely and systematically;allowing,eventually,the most sophisticated techniques to be 'imported',before a dialogue investigation is commenced.There is also potential for this kind of activity in several other systems—though,it may be that none of the existing packages combines flexibility and analytical scope adequately for the applications sketched in Section 15.One recent development that may play an important part in making analytically complex investigations feasible(in the fairly short term)is the design of MODLISP(Jenks(1979)),an extension of LISP intended to facilitate the construction of a modular system that is easily enlarged to accommodate extra facilities and changing environments. Presumably, this may be adapted for use on the parallel program,parallel data LISP machine now under construction at the University of Utah,which will enhance the facilities for symbolic analysis still further.In particular,MODLISP can form appropriate, underlying domains 'dynamically',during a computation(rather than 'statically', at compile time),which is of great importance in complex calculations,where the 'most natural' environment amy alter radically from one phase to another.

<u>14.4.2.</u> The basic aim,then,is to give the environment far more structure than has been assumed up to now.If a problem can be formulated sufficiently clearly for closely related background results to be imposed as 'constraints',then its effective solution is facilitated greatly.For,the more narrowly a mathematical field of investigation is defined,the more precisely is the form of 'typical proofs determined for it.In other words,the environment for investigation may be tailored to suit a given class of problems,to a greater extent than seems to have been recognized in the design of current systems.In symbolic analysis,the emphasis is on imitating,as closely as possible,a collaborative dialogue between two(well-informed!)mathematicians,one of whom happens to have a remarkable talent for intricate calculation,often involving many tedious repetitions;and,who never forgets information,unless specially asked to do so.

Every problem has some 'natural location' in the vast superstructure of mathematics.In fact,there is a whole family of such locations,and even this is not determined absolutely;but,all that is necessary is for <u>some</u> location to be associated with each problem encountered.This idea of location is subjective:it reflects the personal approach of the investigator,including as it does the selection of concepts and results thought to be especially pertinent to the investigation at hand--which is(logically)'located' in relation to these results.For a mathematician, the location of any proposition is strongly time-dependent;but,even this aspect can be simulated,provided that new premises(e.g.,recent results in any field,or even changed 'attitudes' to various faces of the calculations)can be incorporated into the system without undue labour.This framework for investigation is particularly appropriate for interactive symbolic computation,since the user is free to impose on the system whatever constraints(in the form of assumptions(e.g.,conjectures,hypotheses),and theorems)are considered important.Thereafter,the system may be used to study the problem(often,under changing constraints,as the investigation proceeds),until a satisfactory conclusion is reached(either as an'approximate solution',if equations or inequalities are involved;or else,as the statement of some result,deduced from the premises with the help of the machine).Naturally, this approach to dialogue computing is adopted already to some degree;but it is possible now to extend it to cover a wide range of mathematical procedures.The demands that such a scheme makes on the designers are considerable,and should not be underestimated;but the rapid progress made in the past decade in the design of both hardware and software augers well for the increased sophistication required. Thus,it is certainly necessary to examine various mathematical fields where the potential for constructive procedures is high(if the additional manipulative power of computers is invoked).This is done in Section 15,and,in some cases,'environmental constraints' are identified,and tentative 'flow diagrams' are given.The aim is to show that symbolic analysis could be a standard research tool,in many areas so far largely remote from computational activity.

14.4.3. As a foretaste of what can be done, this section is concluded with some observations on the use of inequalities, in 'operational form', which enables a user to generate compound inequalities—and these, in turn, could be used in forming error estimates for various approximation procedures (for instance, in the constructive solution of nonlinear operator equations). Before this can be done, however, it is necessary to ask an apparently naive question, the answer to which turns out to be far from trivial, namely: What is an inequality ?

It would appear that inequalities may be studied on many different levels of abstraction. For their use, only the most concrete representations are adequate; but, for classifying them, or investigating the interrelationships among different types of inequalities, more general frameworks are appropriate (especially, if one wishes to use arbitrary, partially ordered sets as a basis, rather than the field of real numbers). For this reason, a basic definition is given now, which, although straightforward, does allow for the possibility of inequalities among the elements in any, suitably-ordered set.

Definition. Let S be any set, and T, any totally ordered set (with order relation, $<_T$). Then: an inequality on S is an ordered quintuple, $(\alpha, \beta, \sigma, \sigma', T)$, where, if Σ_S denotes the set of all countable, ordered subsets of S, then $\sigma, \sigma' \varepsilon \Sigma_S$, and each of α and β is a mapping from Σ_S into T, such that $\alpha(\sigma) <_T \beta(\sigma')$.

(Notice that one could take T to be merely partially ordered—e.g., a lattice. This possibility can be accommodated here, with fairly obvious modifications, which can be formulated as they are required). In the special caseswhere it makes sense for one, or both, of α, β to be (equivalent to) the identity mapping, or for σ' to coincide with σ, various inequalities between (the values of) real-valued functions may be obtained. Again, some familiar inequalities have the form: $[\forall\sigma \varepsilon K \subset \Sigma_S, \forall\sigma' \varepsilon L \subset \Sigma_S]$ $\alpha(\sigma) <_T \beta(\sigma')$; but in each individual instance, the inequality is generated by the basic construction given here. Moreover, S may be finite, or else countably or uncountably infinite: what matters is that appropriate mappings α, β can be defined. In the context of metric or normed spaces, the ordering, $<_T$, may (but need not) take an obvious form; but in other cases, the definition of suitable orderings is often far from clear. However, the essential point is that a general definition may be used to describe (and generate) inequalities of all possible types which could be of importance in symbolic analysis. Inequalities should be incorporated in such a way that their application to specific expressions could be effected through substitution into the general forms. Plainly, all of the standard inequalities of 'hard' analysis are easily accommodated within this scheme (and they would be available for routine use in analytical procedures).

As a very simple example of this idea, let S be $L^2(I)$, for some real interval, I, and let $f \varepsilon L^2(I)$(but regarded as an element of Σ_S) be given. If the system splits f into(algebraic and transcendental)'factors': $f = g_1 \dots g_M h_1 \dots h_N$, then one could define $\alpha(f)$ to be $|\int f|$, and $\beta(f)$, as $\surd\{\Pi\int g_j^2 \; \Pi\int h_k^2\}$, the ranges for the integrations and products being I;1,...,M;and 1,...,N, respectively. From this it is clear that any classical inequality can be handled in this kind of way; and that S could be, for instance, a set of square matrices(for whose elements various inequalities involving their traces, ranks, determinants, etc., may be obtained), and so on. On the other hand, if S is, say, a lattice, over which some extra condition holds(e.g., the 'modular' inequality—see, for instance, Birkhoff(1973)), then this may 'induce' nontrivial inequalities, if the elements of Σ_S are mapped suitably into T. In conjunction with the GROUP system(Cannon(1976)—see Section 11), and with analogous packages for other algebraic structures, the inclusion of order relations other than that for the real numbers offers intriguing possibilities for comparatively abstract operations, and broadens even further the potential scope of symbolic analysis as a research tool.

For most of the applications to be outlined in Section 15, only inequalities between real numbers are used; but the variety of sets, S, whose elements give rise to these inequalities(through the mappings α and β), is very great—and the fact that such mappings may be defined and incorporated in symbolic computation systems is of crucial importance for the effective use of these systems in sophisticated mathematical constructions.

The inclusion of procedures applying inequalities presents no problem in principle, since all tht is required is to name the procedure, its arguments, and all parameters. For instance, one could write: HÖLDER(CTS, I, E_1, E_2; r, s), to cause Hölder's inequality for integrals over a set, I, with positive parameters, $r, s, r^{-1} + s^{-1} = 1$, to be applied to the 'splitting' $(E_1;E_2)$ of some basic expression. The 'output' would be $\{\int |E_1|^r\}^{1/r}\{\int |E_2|^s\}^{1/s}$, simplified as far as possible within the system. Further, the Landau ' O , o , ~ ' notation may be included in this scheme, provided that the conditions(all, as $x \to \alpha$, say): $f(x) = O\{g(x)\}$, $f(x) = o\{g(x)\}$, and $f(x) \sim g(x)$, are formulated in terms of inequalities(or limits). In any case, there are operational rules governing the use of the Landau symbols in most situations; and these rules may be implemented fairly simply. As an example, suppose that E is any function bounded in a neighbourhood of α. Then one has(as $x \to \alpha$);

$$E\int_I E_1 E_2 = O\{\int_I E_1 E_2\} = O\{\text{HOLDER}(CTS, I, E_1, E_2; r, s)\} \text{ ,etc..}$$

If the user has a 'catalogue' of procedures that produce upper bounds for expressions(using ' ≤'),while other procedures give lower bounds(using ' ≥'),then,in a typical calculation,procedures may be applied successively(along with other analytical and algebraic routines)until some acceptable bound is obtained(or else, the calculation is shown to be intractable).In other words,the system may be used to investigate expressions in ways that parallel closely the activity of a mathematician alone--but,with the(possibly,crucial)advantage that expressions too cumbersome for 'hand calculation' may be analysed comparatively fully.Of course,the apt choice of basic procedures can shorten a proof considerably;but, since the machine is not working alone(in contrast to the situation for 'automatic theorem-generation'),this problem-solving aspect of mathematical activity is not lost.The great complexity of expressions(and,often,inadequately simplified output)may limit the effectiveness of this human/machine collaboration;but, in principle,extremely high efficiency is attainable(as simplification routines, and clarity of I/O are improved).Thus,in addition to using the standard inequalities,one can introduce more specialized conditions(e.g.,for various types of stability associated with ordinary differential equations--see,for instance,Reissig, Sansone and Conti(1974)--or,for conditions ensuring the convergence of certain approximation procedures for elliptic partial differential equations-- the so-called coercivity conditions(see,e.g.,Showalter(1977)).The opportunity for innovation in this area is very great.Moreover,if symbolic analysis is used in conjunction numerical computation,then many algorithms of considerable complexity can be applied to obtain approximations to the solutions of difficult problems--and these effective solutions will be produces,initially,in symbolic form(often,with error estimates), so that their analytical properties may be studied,as well as their predictions of numerical values.

The remarks made in the present section about mathematical uses of symbolic computation are intended mainly to indicate possible lines of development.In Section 15,through several nontrivial examples,I try to convey my own vision of how a wide variety of constructive procedures can be implemented(approximately). This entails using allof the strategies outlined so far,as well as others;and,although only tentative procedures can be given,the enormous potential scope of this enterprise should become clear.In terms of the current capabilities of systems,I may be somewhat over ambitious(and,unreasonable in my demands);but I am sure that the great bulk of the schemes I sketch will become feasible in the very near future.

15. Mathematical Applications.

The object of this section is to discuss briefly a wide range of topics for which symbolic analysis routines could be developed.The treatment of these topics is very uneven,depending,amongst other things,on:how 'uncomputational' they are at the moment;how far I have studied them;and,to what extent current systems seem able to cope with the demands that nontrivial calculations would make.However,no topic is included for which it is not clear(to me,at least!)that substantial applications—immediate,or 'strongly potential'—have been identified.Thus,in some cases,a fairly definite scheme is described(on which experimental programs can be based);while,in others,no attempt is made to do more than give a bare outline of the topic,sufficient to indicate the sort of calculations that could be done(possibly,on systems of the next technological generation).Of course,the list of 'suitable cases for treatment' is virtually endless:every mathematician with a constructive turn of mind could add to my selection of topics(and may disagree with some of them).My aim is simply to encourage the use of effective mathematical procedures—and the development of such procedures in areas where they are rare. Symbolic analysis provides a powerful incentive for this activity.

Before the examples are discussed,it is necessary to explain,as clearly as possible,the sense(s) in which 'solutions' are obtained,with the aid of symbolic analysis.In problems where algebraic manipulations,expansions into finite sums,and other exactly realizable operations suffice,a correspondingly exact solution is to be expected.Very few of the calculations mentioned here are of this type (but it should not be imagined that such calculations are trivial;on the contrary, they make high demands on both programming skill and simplification facilities). Next,there are problems where most of the steps in the solution can be performed exactly,but a few must be approximated in some way.Typically,this involves the truncation of infinite sums or the use of a quadrature formula to evaluate integrals.In most of these cases,reliable error estimates(in symbolic form)can be given,and the solutions obtained are fairly complete,for either analytical or numerical purposes.Lastly,there is a whole spectrum of calculations in which—to varying degrees—essential components can be tackled only by invoking a succession of diverse approximation procedures,for many of which the determination of symbolic error estimates is a major problem(as is the ultimate justification of the calculation as a whole).Most of the applications discussed here are in this category; but,in several instances,rigorously derived algorithms are available,often,of great subtlety and complexity;and it is on these algorithms that the treatments given here are based.The crucial point is that,under various conditions,which must be adumbrated,adequate symbolic approximations can be obtained,even after a whole range of approximation procedures have been used at intermediate stages—provided that theconvergence at each stage is 'strong enough'(relative to a suitable topology).In certain examples,these questions are not resolved completely,but all gaps that remain are identified.

15.2. Implicit functions of one complex variable.

Let F: $D \to C$ be any function defined on a domain in CxC(the cartesian product of the field of complex numbers with tiself).This function may be studied for its analytical properties,e.g.,various types of continuity or differentiability;but,it may be used,also(in the form of an identity,$F(z,w) \equiv 0$ over D)to define 'implicit functions',W(z),or Z(w),through the relations: $F(z,W(z)) = 0$ ($\forall z \in D_1$),and $F(Z(z),w) = 0$ ($\forall w \in D_2$),respectively,where D_1 and D_2 are domains in C.This problem, which,for real-valued functions of two real variables,is treated in basic analysis courses,assumes great importance when complex variables,and complex-valued functions are considered.If F is arbitrary,then very few useful results are obtainable.However,for several classes of functions F,the theory is extensive, and covers many significant types of implicit functions occurring in mathematical physics and other fields.

For this paper,a convenient starting point is the so-called Weierstrass Preparation Theorem(see,e.g.,Saks and Zygmund(1971);and,for a more abstract treatment, Grauert and Fritzsche(1976)).This provides a factorization in the form:

$$F(z,w) = (z-z_0)^p F_1(z,w) \sum_{0 \leq r \leq k} A_{k-r}(z)(w-w_0)^k \qquad (*)$$

where F is holomorphic,and does not vanish identically,in a bi-circular neighbourhood of the (finite) point,(z_0,w_0),$F(z_0,w_0) = 0$,and the relation (*) holds in some(generally,smaller),bi-circular neighbourhood,say, $\Delta(z_0;\rho)\times\Delta(w_0;\rho)$—over which $F_1(z,w)$ is nowhere zero.Moreover,the 'coefficient functions',$A_j(z)$,are holomorphic in $\Delta(z_0;\rho)$. For the rest of this subsection,it is assumed that such a decomposition is possible,in all situations considered.(For computational purposes,one should note that,when $F(z_0,w)$ is not identically zero in a neighbourhood of w_0, the integer,p,in formula (*) is 0,and k is the multiplicity of w_0 as a root of the equation $F(z_0,w) = 0$.Further,replacement of $z-z_0$,$w-w_0$,respectively,by z^{-1}, w^{-1},allows (*) to be used when z_0 or w_0 is ∞.

If F has the decomposition (*),the A_j being general holomorphic functions,and if $p = 0$,in (*),then the identity $F(z,w) \equiv 0$ (over D)determines algebroidal functions, W(z),or Z(w);see,e.g.,Rémoundos(1927),Selberg(1934).If,however,the A_j are polynomials,then algebraic functions are determined.In either case,the identity defines k holomorphic function elements in a neighbourhood of z_0(the branches of the k-valued function corresponding to (*));and,such functions are associated with Riemann surfaces--over which they behave as single-valued analytic functions(see, e.g.,Springer(1957)).Accordingly,the following problems appear suitable for computational investigation(using symbolic analysis).

Problem 1. Given (*),find the essential characteristics of the Riemann surfaces for the functions W and Z.

Problem 2. Develop an algorithm for the effective representation of the branches of Z and W.

Problem 3. Apply the results of Problems 1 and 2 in 'interesting cases'(e.g., in the determination of expansions for various transcendental and algebraic functions of interest in practical situations).

For Problem 1,the crucial matter is to find all critical points of W(or,of Z), where α is critical for W if at least one function element,W_j,is not arbitrarily continuable in some neighbourhood of α.This problem has been studied extensively(see,e.g.,Walsh(1950)).Of special significance for Riemann surfaces are the branch points,whose locations and orders essentially fix the structure of the k-sheeted surface corresponding to (*)--see,for instance Forsyth(1918),for many detailed,specific examples.Thus Problem 1 may be formulated as follows:to develop effective methods for the (approximate)determination of critical points,to find the nature of these points,and to use this information to give a concise description of the Riemann surface so determined.

Notice that,in this procedure,the type and order of the critical points must be determined exactly:the approximation--if there is approximation--refers only to the location of the critical points,since this may be determined as a zero of a transcendental function,or in some other way that precludes exact calculation. The necessity for approximation does not,however,affect the characterisation of the topological properties of the Riemann surface.

Problem 2 has been studied from the angle of computational complexity by Kung and Traub(1978),for algebraic functions.They mention several possible extensions, including a general use of Puiseux series(fractional power series),and the case where the coefficients of the expansion series are expressed explicitly as functions of the input coefficients.No definite results are given for these and other extensions,but it is certain that symbolic computation can play a basic rôle here.The most important aspect of this work is that 'fast algorithms' are given for general algebraic functions,reducing the worst-case result of $O(N^k)$ 'elementary operations',for computing the first N coefficients in the expansion of a branch,W_j,of W,to $O(kM(N))$,where M(N) is the minimal number of operations sufficient for computing the product of two polynomials of degree N(over a specified ground field).For direct multiplication,$M(N) = O(N^2)$,but this may be decreased to O(N log N),if the Fast Fourier Transform is used(see,e.g., Knuth(1969)).Kung and Traub study various types of iteration to find the unknown coefficients,and they combine this with a symbolic version of the 'Newton Polygon' scheme (see,e.g.,Bliss(1932)),which can be implemented effectively within symbolic computation systems. Although simple cases are treated by Kung and Traub,the algorithm they give--or an adaptation of it--should be generally applicable.Thus:fast,effective,symbolic computation of expansions for branches of general algebraic functions is possible--and procedures could be developed for several of the systems considered in this paper.The most urgent matter for applications is to provide similar facilities for algebroidal functions;which should not present major difficulties.

Problem 3. It is easy to see that the Nth root,reciprocal and reversion of a polynomial are obtainable in terms of expansions of algebraic functions(corresponding results for power series would involve algebroidal functions.Again,all of the inverse Elementary functions could be treated on the basis of algebroidal functions;and,Kung and Traub show how Special functions may be treated(using the algebraic conditions satisfied by their generating functions)--and that elliptic integrals may be included,too.Even without a general procedure for algebroidal functions,certain transcendental equations may be treated(when the transcendental terms have known expansions).Another important application is to computational algebraic geometry(which has become central to the symbolic integration of algebraic functions--see,especially,Davenport(1979a,b)).More general extensions,in which (one,or both of) z and w are finite-dimensional vectors,occur in one approach to branching phenomena for nonlinear operator equations,where(an extension of) Newton's Polygon method is used--see Section 15 .It will be obvious from the examples given in the present section,that calculations with algebraic/algebroidal functions have exceptionally wide application,so no more instances will be cited here.Further,the extensions(to Puiseux series,vetor variables,finite ground fields,fully symbolic input)mentioned by Kung and Traub(1978)are met in a great variety of contexts;so,the basic aim is to develop general procedures that canbe adapted,as required.For functions outside the algebraic/algebroidal class,only isolated,special methods are known,and these are,mostly,unsuitable for symbolic analysis.

15.3. Computational function theory.

Many techniques in function theory are capable of effective formulation.Obvious examples are:manipulations with power series(see Brent and Kung(1978)for a complexity analysis,covering composition,reversion,evaluation of Elementary functions 'at power series arguments';solution of certain differential equations with power series coefficients —even some nonlinear equations,using iteration;evaluation of hypergeometric and Bessel functions of power series;multivariate series);calculations with analytic continued fractions;asymptotic analysis of Special functions, and of integral transforms;and,solution of ordinary differential equations,using contour integrals.Other topics found to be suitable for symbolic computation include the solution of differential equations,using expansions in Chebychev polynomials(Geddes(1977)),and calculations involving Padé approximants(Geddes(1978)). Moreover,the basic theorems of local function theory(and some of their direct applications),may be given constructive forms.A great variety of constructive procedures,treated from the algorithmic point of view,may be found in the books by Henrici(I(1974),II(1978),III(in preparation)).It is enough here to observe that many of Henrici's procedures are suitable for effective implementation in symbolic computation systems(for instance,a method for analytic continuation along a curve, unified treatment of Special functions in terms of hypergeometric functions, zero-determination for polynomials,and function theoretic representations for the solutions of certain partial differential equations).Moreover,even some of the more abstract constructions(see,e.g.,Tsuji(1959),Goluzin(1969))are accessible to effective treatment using symbolic analysis(for example,certain extremal problems and 'distortion theorems' arising in the theory of conformal mapping,calculations involving harmonic measures and capacities—especially in relation to approximation problems).See also,Garnett(1972);and Smirnov and Lebedev(1968),where a wide range of approximation problems in constructive function theory is considered. In all of these instances,the aim is to obtain approximations in symbolic form for all objects of primary interest.Although these approximations may be far from optimal where numerical results are required,there are many situations in which only analyticalrepresentations are relevant—since,the aim is to study the properties of functions obtained by using these representations in earlier phases of calculations(see,e.g.,Section 15 on boundary value problems in elasticity).It is precisely in areas of this kind that the interplay of symbolic and(ultimate) numerical computation is so important.One particular problem where symbolic analysis may be able to play a major part stems from Levinson's(1974)attempts to verify the Riemann Hypothesis by direct estimation—essentially using Rouché's theorem,in the form of the 'argument principle',certain 'mollifier'functions being introduced to smooth out irregularities.By using symbolic analysis,it may be possible to test many different mollifiers,and so to improve Levinson's result (that 'more than one third of the zeros of the Zeta function lie on the 'critical line', $\{ z \mid Re(z) = 1/2 \}$).

15.4. Calculations involving Patil's(and analogous)representation(s).

Possible alternatives to the classical theory of functions are furnished by certain representation theorems for functions analytic in the Unit Disc,D_1,and satisfying various conditions on parts of the boundary, ∂D_1.One formula of this kind was found by Carleman,and extended by Goluzin and Krylov(1933) in connection with problems of analytic continuation.Other formulae are given in Zin(1953),Picone (1954) and Patil(1972).Here,attention is confined to Patil's work,and some of its immediate consequences.

Patil proves(using functional analytic techniques,involving Toeplitz operators) that if,for $1 \leq p \leq \infty$, $f \varepsilon H_p$(the so-called Hardy class of functions,f,regular in the interior of D_1,and such that ,as $r \to 1^-$, $\int_0^{2\pi} |f(re^{i\theta})| d\theta = O(1)$--uniformly in θ),then f(z) may be found,for z interior to D_1,from the boundary values of f on any set $E \subset \partial D_1$,provided only that E has positive(one-dimensional)measure.More precisely,if $h_\lambda(z) := \exp\{ -(1/4\pi)\log(1+\lambda) \int [e^{i\theta} + z][e^{i\theta} - z]^{-1} d\theta\} (z \varepsilon D_1)$, and $g_\lambda(z) := (K_\lambda h_\lambda)(z)$,where K_λ is the 'weighted Cauchy operator',defined by:

$$(K_\lambda h_\lambda)(z) := \lambda h_\lambda(z) + \frac{1}{2\pi i} \int_E (w-z)^{-1} \overline{h_\lambda(w)}\, \gamma(w)\, dw,$$

the bar denoting complex conjugation,and γ the'boundary function'for f--defined over E,in the present case,then ,as $\lambda \to \infty$, $g_\lambda(z) \to f(z)$,uniformly,on compact subsets of(the interior of)D_1--and,if $p \not\leq \infty$,then $\| g_\lambda - f \|_p \to 0$.

The idea of such a representation is suggested by the result(see,e.g.,Duren(1970)), that, if $f \varepsilon H_p$,$p \geq 1$,and $f(e^{i\theta}) = 0$,on a θ-set of positive measure,then f(z) vanishes identically over the interior of D_1.Another representation corresponds to Privalov's uniqueness theorem:if f is analytic for $|z|<1$,and the nontangential limit of f(z),as $z \to e^{i\theta}$,is 0,over a θ-set of positive measure,then f(z) is 0 at all points interior to D_1.

For possible applications of Patil's(or an analogous)formula,one seeks analytic approximations to $g_{\lambda}(z)$ for large,positive values of λ.Although the basic formula refers only to subsets,E,of the Unit Circle,E may be disconnected,having any number of components.Moreover,the procedure may be used in conjunction with conformal mapping,in some cases,to produce more general results.Consequently,the technique outlined here may be combined with various methods for approximating conformal mapping functions(see Section 15. ,for some of these methods —especially,in relation to the mapping of general,n-connected domains onto 'canonical domains').Young(1974)shows that both the Riemann Hypothesis and the Goldbach Conjecture may be given new formulations in terms of Patil's representation(and that the subset,E, may be chosen so as to exclude the 'minor arcs',as used in the original work of Hardy and Littlewood(1920);see,also Ayoub(1963)).In a less specialized context,it seems likely that asymptotic representations of this kind will be useful in studying boundary-value problems where the boundary behaviour is specified only on part of the boundary--or,perhaps,where the boundary function has singularities.Thus,there may be nontrivial links of this approach with methods for solving certain types of 'ill-posed problems'(see Section 15.10).

Young(1974)gives an elementary account of Patil's formula for the special case where E comprises a finite collection of disjoint arcs: $E = \cup \chi_j$,and f is regular in a domain containing D_1.By defining $w_j(z) := c + i \log((z-a_j)/(z-b_j))$,and then,$W(E,z) := \pi + \Sigma_j w_j(z)$,where a_j,b_j,are the end points of the arc χ_j,one constructs a function,W,whose real part equals 0,on E,and equals π,on E^*--the complement of E relative to ∂D_1.If the ordinary Cauchy integral formula(with ∂D_1 suitably indented around the points a_j,b_j)is used to represent $f(z)\exp\{-kW(z)\}$, for z interior to D_1,then it follows that f(z) may be written in the form:

$$f(z) = I_k(E,f,z) + I_k(E^*,f,z) = I_k(E,f,z) + O\{\exp[-k \operatorname{Re}(\pi - W(z))]\},$$

where $I_k(A,f,z) := (2\pi i)^{-1} \int_A (\zeta - z)^{-1} f(\zeta) \exp\{-k[W(\zeta) - W(z)]\} d\zeta$.Since it may be seen that $0 < \operatorname{Re}(W(z)) < 1$,at interior points of D_1,the proof of Patil's formula for this case is complete(with k replacing the continuously varying parameter, λ). Moreover,if the points a_j,b_j,are specified,the order of the error term(as $k \to \infty$) may be determined explicitly.If the asymptotic formula for f(z) is differentiated n times 'inside the integral sign'(which <u>can</u> be justified),then the formula

$$f^{(n)}(z) = \frac{1}{2\pi i} \int_E f(\zeta) \, (d/dz)^n [(\zeta - z)^{-1} \exp\{-k(W(\zeta) - W(z))\}] \, d\zeta + o(1)$$

is obtained(for $k \to \infty$),where,as before,the o(1) term may be determined explicitly in various cases.There are many opportunities for using symbolic analysis here.

15.5. Summability procedures;tauberian theorems;asymptotic analysis.
Although the range covered by the topics in this subsection is very wide,from the viewpoint of symbolic analysis,they have much in common;so it is appropriate for them to be considered together.Essentially,all of the methods mentioned here amount to the application of various transformations(on sequences);or else, of certain'operational rules',with associated analytical conditions(usually,involving the asymtotic behaviour of sequences,or of more general functions).The potential applicability of symbolic analysis to this area will be evident from an examination of the references cited later on;but the variety of possible procedures is so great that there is no point in singling out particular examples.

It is well known that,for a wide class of functions on real domains,several types of 'summability' may be defined.The standard summation of series,and the evaluation of(Riemann or Lebesgue)integrals may be accomplished only for relatively restricted classes of functions(series corresponding to countable sequences—which are just functions on(subsets of)the set of positive integers).Consequently,methods have been developed for attaching(unique)to the results of summation operations involving more general classes of elements.There are many procedures available for sequences and series(sometimes yielding identical results),but only a few for handling(classically,divergent)integrals.Mostly,a method of summation for series may be characterized(at least,partially),by some asymtotic condition(s)--among these being the so-called tauberian conditions.However,the detailed specification of interesting summation methods often requires the use of (infinite) matrices;and the potential use of symbolic analysis in this area hinges on this mode of representation.(See,e.g.,Petersen(1966);and ,for general accounts of summability methods,Knopp(1928),Moore(1938).Hardy(1949)discusses some delicate analytical questions,and Cooke(1950)gives a general treatment of infinite matrices and their associated sequence spaces.Fairly recent references,and an account of the principal tauberian theorems for summability,may be found in Peyerimhoff (1969)).
The use of general summability methods in applications is clear:it allows rigorous results to be obtained(for subsequent use)in situations where the classical techniques are inapplicable.The rôle of tauberian theorems is to extract information about mathematical objects from various 'partial averages' of them.The tauberians conditions built into a symbolic computation system may be applied in any problem where such partial averages are under consideration--the 'output' usually being some 'dual' asymptotic condition—either,as a final result,or else, for later use in a complicated calculation.Thus,the broad aims for symbolic analysis in this domain include:to incorporate various summability procedures for (approximate)implementation;to allow the evaluation of certain(classically

divergent)integrals;and,to examine the use of a wide range of tauberian theorems in 'operational form'(see Section 14).

In this paper,remarks are confined to the most easily implementable schemes: matrix limitation methods(on the vector space,S^*,of all real sequences).The field of definition of a linear transformation,$\hat{\tau}$,on sequences,is a vector subspace,say,D_τ,of S^*.The convergence field,C_τ,of $\hat{\tau}$,is defined as the set of all σ in S^* such that $\hat{\tau}\sigma$ is classically convergent.(Thus,in an obvious notation,$C_0 = S^*$, and $C_1 = C^*$--the subspace of all classically convergent sequences).When $\sigma \varepsilon C_\tau$,and the sequence $\hat{\tau}\sigma$ converges(classically)to t,one says that σ is $\hat{\tau}$-limitable to t. A regular linear transformation is one that acts'naturally'on classically convergent sequences;i.e.,for such a transformation,$\hat{\tau}$,one has $C^* \subset C_\tau$,and every $\hat{\tau}$-limit of an element of C^* coincides with its C^*-limit.

Notice that,in spite of the fact that the representing matrices are infinite, limitation methods can be useful in symbolic analysis.This is because:(i)necessary and sufficient conditions for the existence of the transformed sequences are known,and involve only concepts of ordinary convergence(for which several tests may be implemented--e.g.,using a LIMIT facility);(ii)the matrices for well-known methods(Hölder,Nørlund,Riesz,...),have specified elements,and so may be incorporated for symbolic computation(in the sense that any given number of terms may be taken in evaluating the matrix operations corresponding to infinite matrices). Consequently,arbitrary'finite sections' of all relevant matrices may be used to form approximations,when the summability methods are applied;and,since all of these methods depend,ultimately,on the classical notion of limit,in principle,any desired degree of approximation can be attained(within the constraints of the size of the computer used,etc.).
For example,if $\{p_k\}$ is a sequence of nonnegative numbers,and $p_1 > 0$,then the Nølund mean, $\{t_m\}$,of $\{s_n\}$,is defined by $t_m := P_m^{-1}(p_m s_1 + p_{m-1}s_2 + \dots + p_1 s_m)$, where $P_k := p_1 + \dots + p_k$.The corresponding limitation matrix has elements a_{mn}, given by: $a_{mn} = P_m^{-1}p_{m-n+1}$,for $n \leq m$,and$=0$,otherwise.For the Riesz mean,one has, instead,the results: $t_m = P_m^{-1}(p_1 s_1 + p_2 s_2 + \dots + p_m s_m)$;and,$a_{mn} = P_m^{-1}p_n$,for $n \leq m$, and $= 0$,otherwise.Many interesting forms of sequential behaviour can be handled by using suitable summability methods;and the approximate implementation of such methods is quite feasible in symbolic analysis.In the present context,theorems which prove that 'generalized summability plus extra conditions implies classical convergence',are called tauberian theorems(the 'extra conditions' being tauberian conditions).This terminology is used more generally,too,e.g.,in relation to asymptotic analysis,where no questions of summability,as such,are involved(see,e.g., Pitt(1958),Wiener(1933),and,for a more abstract formulation,Loomis(1953)).The scope for using diverse tauberian theorems in symbolic analysis is wide,provided that these theorems are treated as indicated in Section 14.

For the 'summation' of(classically divergent)integrals,fewer methods are available. Those most widely used determine the Cauchy principal value(see,e.g.,Levinson and Redheffer(1970)),the finite part ,and the logarithmic part(for both of which, see,e.g.,Bureau(1955)).One further type of summation that may be implemented fairly directly is that corresponding to asymptotic series(in the sense of Poincaré), for which a formal calculus exists.All of these techniques,together with many devices for obtaining the asymptotic developments of functions defined in various ways(e.g.,as integrals with 'large parameters',or as solutions of(systems of) differential equations),are very suitable for the intelligent use of symbolic analysis.For instance,fairly general routines could be implemented to produce,as 'output',asymptotic expansions of functions specified(implicitly or explicitly) as 'input'.This kind of routine would be of great value for calculations having a numerical phase.A good reference for asymptotic procedures on this level is Sirovich(1971),which includes many routines potentially adaptable for symbolic computation.The theory and application of singular perturbations is also potentially adaptable for the use of symbolic computation,in systems where advanced analytical facilities are available;and this is especially important in connection with the asymptotic solution of nonlinear boundary value problems for models of practical interest.See,e.g.,O'Malley(1974).

15.6. Differential calculus in general spaces.

Several notions of 'differential',and of 'differentiation',have been defined for finite-dimensional and infinite-dimensional linear spaces.The most familiar of these are probably the Gâteaux and Fréchet differentials(with corresponding derivatives),but there are many more.The spaces for which such objects can be defined include Banach spaces,and even linear topological spaces.There is no need to give details here(beyond some basic definitions),since the comprehensive review by Nashed(1971)—on which most of this subsection is based—contains a very large bibliography;while,in the contiguous area of 'generalized inverses', the conference proceedings edited by Nashed(1976) contains an annotated bibliography of over two hundred and fifty pages!The object of this subsection is just to identify some uses of calculus in abstract settings where the(potentially)algorithmic nature of the calculations would allow some procedures to be implemented for symbolic computation.Examples of such procedures include:evaluation of Frechet,Gateaux(and other)derivatives—with representations of the corresponding differentials (as linear mappings);evaluation of functional derivatives,using routines suitable for applications(e.g.,in statistical mechanics and quantum field theory:see,for instance the paper by Stell,in Frisch and Lebowitz(1964), and Visconti(1969));approximate solution of various optimization problems(e.g., in control theory or mathematical economics);and determination of implicit functions defined by nonlinear operator equations.(Certain applications of the abstract techniques to problems in concrete,'hard',analysis have been made,too;but these are not discussed here,since they are,for the most part,not amenable to algorithmic presentation).Differentials play a basic rôle in the study of nonlinear mappings between general spaces(especially in the approximate solution of the corresponding nonlinear euqations,using iterative methods).Routines for this class of calculations may be covered,too.There are several cases to be considered,and these should be treated separately,as far as computation goes.However,the fundamental definitions may be formulated quite generally:it is in the specification of norms (or topologies) that the distinctions arise—and it is these possible choices of norms,etc.,that may be programmed for symbolic computation,with provision for basic operations(such as'differentiation' of composite functions,'differentiation along a subspace',and the formation of higher-order derivatives and differentials).As in the case of summability procedures,many of the generalized calculus operations and their applications may be formulated in an essentially finite-dimensional context,so that effective algorithms for approximation can be developed. It is in this sense that all of the following remarks should be interpreted.

Let I be an open interval in R(the real line),and $\Phi : I \to X$,where X is a normed linear space over R.Then the derivative of Φ at $t_0 \in I$ is defined to be the limit, as $t \to t_0$ of $(t - t_0)^{-1}(\Phi(t) - \Phi(t_0))$.When this limit exists,it is denoted by $\Phi'(t_0)$--the sense of the limit being that the norm of the difference of the two elements tends to 0,as $t \to t_0$: $\| \Phi'(t_0) - (t-t_0)^{-1}(\Phi(t) - \Phi(t_0)) \| \to 0$.The Gateaux variation , $\delta F(x_0;h)$,of a mapping $F:X \to Y$,with increment h,is defined as $\lim_{t \to 0} t_{-1}(F(x_0 + th) - F(x_0))$;and this definition is meaningful even if X is not normed.Moreover,if $\delta F(x_0;h)$ exists at all,then so does $\delta F(x_0;\lambda h)$,for all real λ. It may be shown that,if $F:X \supset U \to Y$ and $x_0 \in U$,then $\delta F(x_0;h)$ exists IFF whenever h is in X and $h + x_0$ is in U,one has $F(x_0 + h) - F(x_0) = H(x_0;h) + r(x_0;h)$,where $r(x_0;th) = o(t)$,as t tends to 0,and H is homogeneous of degree 1 in h.If,in addition, $\delta F(x_0;h)$ is linear and continuous in h,then the convention is to write $\delta F(x_0;h) := DF(x_0;h)$,and to call this the Gateaux derivative of F at x_0.Variants of this definition are obtained if '$F(x_0 + th) - F(x_0)$' is replaced by $F(x_0 + \psi(th)) - F(x_0 + \chi(th))$, ψ and χ being operators defined in the neighbourhood of $\underline{0}$ in X,and such that $\max\{\psi(th), \chi(th)\} \to 0$ as $t \to 0$.All of these variants may be covered for symbolic computation,provided that the 'actions' of ψ and χ are specified explicitly.Next,if the condition '$t^{-1} r(x_0 + th) = o(1)$,as $t \to 0$' is replaced by:' $\| r(x_0;h) \| = o(\| h \|)$,as $h \to \underline{0}$ ',and if $H(x_0;h)$ has the form $H(x_0;h) := dF(x_0;h) = L(x_0)h$,where $L:X \to Y$ is linear and continuous,then $F'(x_0):h \to dF(x_0;h)$ is called the Fréchet derivative of F at x_0.If $F'(w)$ exists, for w in $W \subset X$,then the operator, $F': W \to \mathcal{L}(X,Y)$ is called the Fréchet derivative of the operator F.(Here,$\mathcal{L}(X,Y)$ denotes the space of linear operators on X into Y). Since both of these derivatives,and others,to be mentioned,are defined(as they must be) ultimately in terms of limits of quotients of real numbers--these real numbers being themselves determined by basic analytical/algebraic operations,depending on how the norms are defined,etc.--their(approximate)representation,and manipulation for symbolic analysis is quite feasible,in terms of the LIMIT operation,and others.Indeed,it is this relatively sophisticated analytical capability that allows the use of symbolic analysis in most of the diverse applications discussed in Section 15;though there are a few cases where purely algebraic operations suffice.

Clearly,if one defines $\Phi(t)$ as $F(x_0 + th)$,then $\Phi'(0)$ coincides with $\delta F(x_0;h)$. Weakertypes of'variation' and 'derivative' may be obtained,for instance,by using the 'Schwarz derivative'(analogous to the Cauchy Principal Value integral),defined by: $\Phi^{<\prime>}(0) := \lim_{t \to 0} (2t)^{-1}(\Phi(t) - \Phi(-t))$,instead of $\Phi'(0)$.The following points are worth noting.(a).The Fréchet differential is invariant to equivalent changes of norm in X or Y;so that,in particular,all possible choices of norms,when X and Y are both finite-dimensional,produce the same collection of Fréchet differentials (and the norm most convenient for calculation may be chosen at all stages).

(b).The most general differential that obeys the 'chain rule',and reduces to the elementary form of differential for real-valued functions of one real variable, is the so-called Hadamard differential,say, $\Delta F(x_0) := \lim_{t \to 0^+} (F \circ g)'(t) =: Lg'(0^+)$, for the ordinary derivative,',of the composition of F and a suitable function,g, such that $g(0) = x_0$,and $g: [0,1] \to X$ has a derivative,g',for which the limit as t tends to 0^+ exists.The existence of the Hadamard derivative (i.e.,of L) is equivalent to the representability of $F(x_0 + h) - F(x_0)$ as $L(x_0)h + r(x_0;h)$,with $r(x_0;th) = o(t)$,as t tends to 0. (c) Other notions that arise quite naturally in applications include: equi-differentiability (where the limits defining differentials are attained uniformly over families of functions);differentiation along a subspace(a generalization of standard,partial differentiation);and,potential operators(modelled on the classical relation, $df(x;\underline{h}) = \nabla f.\underline{h}$,and defined by: $df(x_0;h) := \langle h, \text{grad } f(x_0) \rangle$,where the angular brackets denote an inner product,and,for $f: X \supset U \to R$, grad $f(x_0)$ is the Fréchet derivative of f at x_0.
(d).Many of the notions of differentiability(say,(α),(β),...) satisfy the nonimplication conditions $(\alpha) \not\Rightarrow (\beta) \wedge (\beta) \not\Rightarrow (\alpha)$.Moreover, functions differentiable in some general senses are not even classically continuous!—and there are many unobvious snags and limitations.Nevertheless,once the basic framework for a problem has been specified,the appropriate forms of differentials and derivatives may be manipulated unambiguously,provided that the underlying conditions sufficient for their existence,and their basic properties are incorporated(as'theorems')as part of the 'environment for investigation',as discussed in Section 14.
(e).For mappings $F: X \to Y$ where at least one of X and Y is a(non-normed)linear topological space,the possibility of defining species of differentiability for F hinges on the definition of sets,A,N,comprising,respectively,'approximating maps', and 'neglegible maps'—after which,F may be called A/N differentiable at x_0 IFF $F(x_0 + h) - F(x_0)$ may be written(uniquely) in the form $\hat{\gamma}h + \hat{\nu}h$,where $\hat{\gamma} \varepsilon A$, $\hat{\nu} \varepsilon N$,and $\hat{\gamma}$ is a linear operator.This definition,however,is too general to be of much use,so extra conditions are imposed—mainly,to ensure that the general notions reduce to the corresponding classical ones,for real-valued functions of one real variable.In this way,forms of differentiability are defined which,in certain concrete realizations,may be useful in symbolic analysis.(See Averbukh and Smolyanov(1967,1968) for details.Various results for higher-order differentials and derivatives may be proved,too,along with several 'mean value theorems'(even for some classes of mappings into non-normed topological spaces),and variants of 'Taylor expansions'with'remainders'.

In applications, a central problem is the determination of extrema of linear functionals(and, in more general cases, where one considers mappings into ordered sets different from R). The basic result here(originally proved by Weierstrass, for mappings on subsets of R^k into R^1, and extendable, directly, to mappings into R^1 of compact subsets of normed spaces) is as follows. If U is a compact subset of a normed linear space, and $f:U\rightarrow R^1$ is continuous, then f attains both its supremum and its infimum(with obvious modifications when f is merely—upper or lower-- semi-continuous). The further extensions of this result--suitably reformulated-- to general topological spaces, mitigates somewhat the awkward fact that some of the most 'natural' sets(e.g., convex sets) in infinite-dimensional normed linear spaces are not compact in the norm topology. However, by exploiting weak topologies in reflexive Banach spaces, one can establish the existence of extrema, over bounded, closed, convex subsets, and some notions of finite-dimensional approximation may be introduced--which offers possibilities for developing symbolic computation routines for such problems. Other facets of this work include the use of various 'implicit function theorems', for which the (approximate)determination of the implicit functions may be reduced, often, to problems involving mappings between finite-dimensional spaces(indeed, the Liapunov/Schmidt procedure, as extended for use in solving nonlinear operator equations, is based on this idea--see Section 15.); the study of generalized boundary-value problems in Banach spaces(via reduction to corresponding constrained variational problems; and, the solution of nonlinear operator equations, using a variety of iterative schemes, well-suited to symbolic computation--especially, where extra 'parameters' are involved--but aimed at eventual numerical results.(See, e.g., Rall(1969), where detailed flow-diagrams and programs are given --some of which are of immediate use in symbolic computation, as well as in numerical calculation). Lastly, there are diverse methods for handling nonlinear programming, and other optimization problems, over normed linear spaces. These include iterative schemes(e.g., 'gradient', 'descent', and 'steepest descent' procedures—which may involve the estimation of 'domains of attration'— relative to various norms); least-squares approximation (where 'generalized inverses' play a basic part: see, e.g., Campbell and Meyer(1980), for many detailed, finite-dimensional procedures); and certain problems in optimal control, where constrained optimization in function spaces is required. A unified, algorithmic presentation of much of this material is given by Beltrami(1970). Another rich source of potential effective procedures for symbolic computation in problems bordering on this area, is Karlin(1959a,b), where a wide range of methods of use in mathematical economics and game theory is analysed in such a way that approximation schemes for many types of problems could be developed comparatively simply.

15.7. Constructive conformal mapping of general,n-connected domains.

The problem of mapping a given domain conformally onto another domain has many facets,ranging from the most practical(e.g.,solving the boundary value problems of classical mathematical physics--see,for instance,Bateman(1932)),to the highly abstract(such as the classification of Riemann surfaces--see,e.g.,Tsuji(1959)). Although it suffices,in some contexts,merely to establish the existence of suitable mappings,their detailed structures—especially,in relation to boundary properties—have implications which are not covered at all by existence proofs;and, in practical applications,explicit representations are essential.Consequently, effective methods of approximation have been developed,mostly,with ultimate numerical calculation in mind.However,some of these methods could serve equally well to determine symbolic approximations to mapping functions,in comparatively simple analytical forms(e.g.,using polynomials,rational functions,truncated power-series expansions)and various 'closed forms',from which the relatively simple approximations may be obtained by means of interpolation,quadrature,asymptotic expansion, etc.—all treated symbolically,as far as possible. A fundamental distinction must be made between the cases of simply-connected($n=1$) and multiply-connected($n \geqslant 2$) domains.For 1-connected domains,the basic result is Riemann's mapping theorem, guaranteeing the existence of a function mapping the given domain,D,one-to-one and conformally,onto any other specified,1-connected domain,D',in such a way that a preassigned point in D(and a 'direction' there) is mapped to a preassigned point in D'(with its chosen direction)—provided that each of D,D',has more than one boundary point.The most convenient version of the theorem is obtained by taking for D' the closed unit disc,and some of the proofs(see,e.g.,Bieberbach(1952))could form the basis for constructive schemes,though not for efficient ones.From the viewpoint of symbolic analysis,the crucial results(for determining mappings of general n-connected domains)hinge on necessary and sufficient conditions for the (uniform) approximability of functions of one class by(sequences of) functions in 'analytically simpler' classes.Adequate results of this kind may be found in Davis(1963) and Walsh(1966);more general cases are treated in Smirnov and Lebedev (1968).The possible use of analytic continued fraction representations(e.g.,in the guise of Padé approximants--see,e.g.,Geddes(1978),where ALTRAN is used)should be borne in mind,too.For most of the techniques to be mentioned in this subsection, see Beckenbach(1952),Kantorovitch and Krylov(1958),Gaier(1964),Goluzin(1969),and, Henrici(1979),where the use of Fast Fourier Transform algorithms is emphasized. (See,also,the forthcoming book,'Volume III',by Henrici).

The main result for n-connected domains($n \geqslant 2$)is as follows(see,e.g.,Bergman(1950)): every domain of connectivity n may be mapped conformally(and one-to-one) onto each of eight 'canonical domains--say,$D_1', \ldots, D_8'$,respectively(namely,concentric circular slits annulus;concentric circular slits disc;concentric circular slits plane;radial slits plane;radial slits disc;radial slits annulus;exterior of several discs;and,parallel slits plane).Note,however, that many other 'image domains' may be used in special cases;see,e.g.,Goluzin(1969).Moreover,analytical procedures may be given to determine the corresponding mapping functions in terms of the Bergman kernel functions ,which are,in turn,constructible,for an n-connected domain, Δ ,as $K_\Delta(w,z) := \Sigma \overline{P_j(w)} P_j(z)$--subject to convergence criteria--where $\{P_j\}$ is a relatively complete system of polynomials,mutually orthogonal over $\partial\Delta$ (resp., over Δ)--see,e.g.,Szego(1939).For 1-connected domains(for which the method was developed first),the mapping functions,say,f and g,respectively,have the representations $f(z) = I_{q,z}(\Lambda^2_{\partial\Delta})$,and, $g(z) = I_{q,z}(\Lambda_\Delta)$,where $\Lambda_S := K_S(q,\zeta)/K_S(q,q)$, and $I_{a,b}(H) := \int_a^b H(q,t)dt$. These formulae are obtained from the following variational principles:(A).Of all functions f regular in D,with $f(0) = 0$,and $f'(0) = 1$, the(unique)function mapping D conformally onto the unit disc minimizes $\int_D |f'(z)|^2 dz$; and,for (B),replace the integral to be minimized by $\int_{\partial D} |g'(z)| dz$. Either of these principles (and the resulting representations for f and g) could provide the basis for an approximation procedure suitable for symbolic computation.

In the case of multiply-connected domains,there is still only one 'area',and one 'boundary' kernel function;but the various mapping functions,say, ψ_j,onto the canonical domains,D_j',are determinable from conditions of the form:

$$\psi_j'(w) = \chi_j(w) + \int_S \Gamma_j(\zeta) K_S(w,\zeta) d\bar{\zeta} ,$$

where the χ_j, Γ_j,depend suitably on parameters characterizing the canonical domains. In principle,these relations may be used to generate approximations to the ψ_j , but several types of intermediate approximations would be required--e.g.,for the integrand,or for the eventual integration of the ψ_j'(though,the powerful facilities offered by variants of the Risch algorithm could be used here).The relations for the ψ_j' may themselves be derived from variational conditions of the form(cf., Nehari(1952)):"The function,f,with values $[E_j \circ h_j \psi_j](z; \underline{a})$ maximizes the functional of f',given by $F^*\{f'\} := \left| \int_{\partial\Omega} [H_j \circ \gamma_j](t) f'(t) dt \right|^2 / \iint_\Omega |f'(t)|^2 d\Omega$ ".

Here,the E_j are Elementary functions(mostly,'log'), $\underline{a}$ is a vector containing all characteristic parameters,and the H_j, γ_j and h_j,are all known functions.Although these conditions have the potential for providing schemes to approximate the mapping functions,there are more efficient ways(at least,for n-connected domains, if $n \geqslant 2$) some of which will be mentioned now.

Apart from quasi-practical methods,based on 'relaxation',analogue computation or graphical constructions(see,e.g.,Southwell(1946),Kantorovitch and Krylov(1958)), which may be indispensable if,for example,the boundaries of domains are determined only through experiment,and cannot be 'fitted' by analytical expressions,there are several(interrelated)procedures in which the (real part of the) mapping function is found from a (system of) integral equation(s).All of these schemes are covered by Gaier(1964);but it is worthwhile studying one scheme here,to illustrate a possible implementation for symbolic analysis.A particular application occurs in extensions of the Muskhelishvili/Kolossoff(and,related) method(s) for solving boundary-value problems in plane elastostatics--see Section 15.9 ,for this and other methods.

Let D be bounded externally by L_0,and internally,by $L_1,\ldots,L_{n-1}$,all of the L_j being piecewise smooth,closed curves.Suppose that a function,f,is sought ,mapping D conformally onto the plane with slits,$S_0,\ldots,S_{n-1}$,all of finite length,in such a way that $f(L_j) = S_j$,for $j = 0,\ldots,n-1$,and a preassigned point,z_0,is mapped to ∞. Then it may be shown that,if $u_k(s)$ denotes $\mathrm{Re}\{f(z)\}$,for z on L_k,then the following('Gerschgorin')system of coupled integral equations is obtained for the u_k:

$$u_k(s) = 2\mathrm{Re}\{c(z-z_0)^{-1}\}_{z=s} + (1/\pi)\sum_{j=0}^{n-1}(-1)^{1-\delta_{j0}}\int_{L_j} r^{-1}\cos(n,r)u_j(\sigma)d\sigma,$$

where c is the residue of f at z_0,$k = 0,\ldots,n-1$, δ is the 'Kronecker delta'

If L_k has the parametrization $x = f_k(t), y = g_k(t)$, $0 \leqslant k \leqslant n-1$,for piecewise smooth functions,f_k,g_k,on some interval, $[a,b]$,then,in the general case,where 's' and 't' refer to points on L_j,L_m,respectively(where j and m may be distinct),one obtains the system of equations:

$$u_j(s) = \sum_{m=0}^{n-1}\int_a^b K_{jm}(t,s)u_m(t)dt + \eta_j(s) \text{ ,where}$$

$$K_{jm}(t,s) := [\{f_j(s)-f_m(t)\}g'_m(t) - \{g_j(s)-g_m(t)\}f'_m(t)]/(A^2 + B^2) \text{ ,}$$

$$A := f_j(s) - f_m(t) \text{ , } B := g_j(s) - g_m(t).$$

The structure of a program to implement this scheme for determining f may be indicated, as follows.

Step 1. Specify the L_k as parametrizations of some interval, say, $[a,b]$.
Step 2. Determine the kernels, K_{jm}, from the representations given in Step 1.
Step 3. Set up the system of integral equations for ~~the u_k~~.
Step 4. Specify an approximation scheme for 'evaluating' the integrals in Step 3.
Step 5. Obtain a resulting system of approximating algebraic equations.
Step 6. Solve this system of algebraic equations (to obtain the values of the u_k at a finite number of chosen points of the boundary.
Step 7. Apply some interpolation scheme to determine an approximation to the real part of the mapping function on the boundary.
Step 8. Use a 'complementary integral relation' (and limiting operations) to approximate the complete mapping function within the domain, and its imaginary part, on the boundary.

The storage and manipulation of these entities is fairly straightforward in most of the systems discussed in this paper; thus, Steps 1, 2 and 3 require no comment here, though various care is required to ensure that the most favourable representations are realized. Notice that all of the kernels must be evaluated before the equations can be set up, and that the parametrizations must be specified in a convenient way, etc.. The choice of possible (symbolic) approximations for the integrals is a complicated matter (see, e.g., Krylov (1962)), but, just to illustrate what could happen, let the parameter interval be equipartitioned at P points, $s_0(=a),\ldots$ $\ldots, s_q, \ldots, s_{P-1}(=b)$, and make the trivial approximation $\int_a^b f(t)dt = (b-a/P)\,\Sigma f(s_q)$, the summation over q ranging from 0 to P-1. With these preliminaries, the following system of nP equations in nP variables is obtained for the values of the u_k at the points s_q: $U_{jq} = \sum_m \sum_r K_{jm;qr} U_{mr} + \eta_{jq}$, where, $K_{jm;qr} := K_{jm}(a+P^{-1}q(b-a), a+P^{-1}r(b-a))$; etc.. From the (known) existence of a solution to the original system of integral equations, it follows that the derive system of algebraic equations is (uniquely) solvable. This step is by no means trivial, but all of the symbolic computation systems have highly-developed routines for this procedure, so there is no point in discuusing it further here. The final problem is to retrieve (approximately) the values of f within and on the boundary from the approximations found so far. It may be shown (see, e.g., Kantorovitch and Krylov (1958)) that this may be done by means of the formula:

$$f(z) = c(z - z_0)^{-1} + iv_0 + (2\pi i)^{-1} \sum_{j=0}^{n-1} \int_{L_j} (\zeta - z)^{-1} u_k^*(\zeta),$$

where v_0 is a constant, and u_k^* denotes the approximation to u_k determined in the preceding steps. This representation of $f(z)$ is valid only for points z interior to the boundary: to obtain boundary values, limiting operations must be used. This is quite feasible, with the LIMIT facilities available in the more sophisticated systems. Thus, the whole procedure is potentially viable, with a choice of approximation subroutines for which symbolic error estimate are known.

Symbolic computational treatments of other systems of equations for determining mapping functions may be developed analogously.

A quite different method, based on 'potential theoretic representations in terms of boundary densities', was introduced by Symm(1966,1967,1969), analysed more rigorously by Gaier(1976,1979), and by Henrici(1979). The crux of the method is that, if f maps the 1-connected domain, D, onto the unit disc, and $0 \in D$, then, since $|f(z(\tau))|$ has the constant value 1, it suffices to determine $\phi(\tau) := \arg\{f(z(\tau))\}$, for $\tau \in J$, where the parametrization of ∂D has the form $\tau \to z(\tau)$--so that ϕ, a so-called boundary correspondence function, depends on the parametric representation of ∂D, as well as on f. It may be shown that the solution, say, ξ, of the integral equation

$$\int_J \operatorname{Log}|z(\sigma) - z(\tau)|\xi(\tau)d\tau = \operatorname{Log}|z(\sigma)| \quad , \text{ for } \sigma \in J ,$$

has the unique solution $\xi(\tau) = \phi'(\tau)$. An extension to 2-connected domains has been given by Gaier(1964), and numerical experiments have been performed; but, as yet, there is no effective analogue of this method for general, n-connected domains. Henrici's treatment assumes that $\phi'(\tau)$ has a Fourier serie expansion, and derives a set of recurrence relations for the coefficients, for which an iterative solution is obtained. Even though no direct generalization of the method has been found for n-connected domains, there could well be scope for interesting exploration of this approach, using symbolic analysis--especially, since Gaier(1979) has shown that the existence theorem for the integral equation does hold in the general case.

All of the methods mentioned so far involve linear integral equations(or else, direct representations, e.g., in terms of kernel functions); but there is another technique, based on the use of 'conjugate periodic functions', which leads to Theodorsen's nonlinear integral equation, from which the boundary correspondence function may be determined for the function mapping the unit disc onto a given domain(note the 'reversal of direction' of the mapping). The Theodorsen equation may be put in the form(see, e.g., Ostrowski, in Beckenbach(1952); also, Henrici(1979), for a slightly different form):

$$\phi(\theta) - \theta = -(2\pi)^{-1}\int_0^{\pi} [P(\phi(\theta+\tau)) - P(\phi(\theta-\tau))]\cot(\tau/2)d\tau \qquad (*),$$

where, assuming that ∂D is 'starlike relative to the origin', ∂D has the quasi-polar representation: $w = \rho(\theta)\exp\{i\phi(\theta)\}$, the point $z = e^{i\theta}$, on the unit circle, being mapped by γ, say, into $\rho e^{i\phi}$, on ∂D, so that, ϕ is any function continuous on $[0,2\pi]$ such that $\phi(\theta) = \arg\{\gamma(e^{i\theta})\}$, and P is the single-valued, 2π-periodic function corresponding to the 'full parametric representation', $\rho(\theta) = \exp\{P(\theta)\}$.

The obvious iterative scheme for solving (*),namely,calculation of $\phi_{k+1}(\theta)$ by using ϕ_k on the right side of (*),has worked very successfully;but its rigorous analysis is far from trivial.To give an idae of the sort of results obtained (cf.,Ostrowski,and Warschawski,in Beckenbach(1952)),consider the following pair of theorems which,together,constitute tools for the comparison of the exact solution of (a generalization of) equation (*) with the Nth iterate,and with iterative solutions of discrete systems of equations derived from (*).

(T1).The iterative scheme $v_{k+1} = b + AG(v_k)$,for the solution of the equation $v - b = AG(v)$,where A is a square matrix,v a real vector,and $|Av| \leq |v|$,gives a unique solution,for which $|v_k - v| \leq (d^k/1-d)|v_1 - v_0|$,provided that G is continuous,differntiable almost everwhere,and satisfies the condition $|G'(t)| \leq d < 1$.

(T2).If the equation $\phi(u) - h(u) = -(1/2q)\int_0^q \Delta_{\pm t}(F\circ\phi)(u)\cot(t/2)dt$, (in notation for which the integrand corresponds to that in (*),but with 'P' replaced by 'F'),is solved by a combination of 'discretization' and iteration, and if m,E and T are,respectively,the root mean squares of the functions ϕ_0', h',and $\phi_1 - \phi_0$,relative to the mean $Mf := q^{-1}\int_0^q f(\lambda)d\lambda$, then one has

the inequality: $|\phi_k - \phi| \leq 2(1-x)^{-1}[(m + (1-x^2)^{-1}T)E]^{1/2}$,(d being as in (T1), and with the assumption $|F'(\phi)| \leq 1$,for all values of ϕ).Plainly, there is an opportunity for the use of symbolic analysis in this scheme.

It is well known that,for a finite,n-connected,plane domain whose boundary is the union of n disjoint Jordan curves,the Green's function,G,exists(see,e.g.,Tsuji(1959)). Moreover,the conformal mapping function,g,of the domain onto a radial slits domain is linked with G by the relation: $Re\{\log g(z)\} = 2\pi G(x,y)$,for $z := x + iy$;and this,too offers possibilities for approximation procedures which could be apt for symbolic analysis(see,e.g.,Kantorovitch and Krylov(1958),for one simple scheme; also,Mikhlin(1957),for a useful dicussion of Green's functions,and of the generalized Dirichlet problem,for n-connected domains --on which effective procedures might be based).In Gaier(1964),many methods are treated in detail,with error estimates--some of the analyses being suited excellently to the development of symbolic computation routines,in which the(symbolic)error terms are retained,and revised, throughout each calculation.An application of finite element mehtods to conformal mapping may be found in Weisel(1979),where mapping problems are formulated as (singular) variational problems,and studied with the methods of Ritz and Galerkin, in a Sobolev space framework(including error estimates).

Next, mention should be made of 'function theoretic, iterative techniques'(see, e.g., Gaier(1964), Chapter V), the common theme being the substitution of n successive mappings of (in general, different) 1-connected domains, for the mapping of a given n-connected domain. Although these methods—due to many different people, among them, Koebe, Komatu, Hübner, Landau, Grötzsch, Goluzin, and Gaier himself—are not efficient for numerical calculation (at least, for n=2), their strongly function-theoretic orientation, and the consequent rigorous error estimates, suggest that they may be capable of adaptation to symbolic analysis (again, with many intermediate approximations), even for general n—i.e., allowing n to be specified, along with any n-connected domain to be mapped. The various schemes, and their associated error estimates, are somewhat intricate, so no details can be given here; but the analysis presented by Gaier is very adequate for further exploration.

Finally, in this subsection, variational properties are summarised characterizing conformal mappings of typical k-connected domains onto the canonical domains, D_1', ..., D_8', as defined above. See Tsuji(1959) and Goluzin(1969), for more details. Although it is not obvious how these results should be used, they could form a basis for interesting symbolic analytical studies. First, let D^* be an unbounded, k-connected domain, with boundary (Jordan) curves, $C_1, \ldots, C_k$, and let the univalent, meromorphic function, f, have the representation: $f(z) = a_0 + a_1 z^{-1} + \ldots$, in a neighbourhood of $z=\infty$. (In the interests of simplicity of statement, 'max' and 'min', are used, sometimes, instead of the strictly correct, 'sup' and 'inf'). The expansion of f(z) near z=0 need not be specified. The results will be labelled as propositions, P1., ..., P6

<u>P1</u>. Let $f(\infty) = 0$. If D^* is mapped conformally on D_8' by f, then $\mathrm{Re}\{a_1\} \geq A/2\pi$, where A is a positive lower bound on the total area enclosed by the curves C_j. Moreover, that function f for which a_1 has maximal real part maps D^* onto D_8', with slits parallel to the real axis.

<u>P2</u>. Let $f(0) = 0$. If f maps D^* conformally onto D_3'(resp., D_4'), then $|f'(0)| \geq 0$ (resp., ≤ 0). In each case, equality obtains only for the identity mapping.

<u>P3</u>. Let A be as for P1, $J := A/2\pi K^2$, where $K := \max\{\,|z| : z \in \cup_j C_j\,\}$. If f maps D^* conformally onto D_3' (resp., D_4'), then $|f'(0)| \geq e^J$ (resp., $\leq e^{-J}$). Further, if $f(0)=0$, then the mapping f for which $|f'(0)|$ is maximal (resp., minimal), takes D^* conformally onto D_3' (resp., D_4').

The next results refer to the conformal mapping of a domain, say, D, onto bounded, k-connected, canonical domains(i.e., onto D_1', D_2', D_5' and D_6'). It is assumed that D is contained in the closed unit disc, that $0 \varepsilon D$, and that the unit circle and a finite number of continua within it costitutethe boundary of D.

P4. If f maps the origin and the unit circle into themselves, and if f also maps D conformally onto D_2' (resp., D_5'), then $|f'(0)| \geqslant \exp(\alpha/2\pi)$--hence, also $\geqslant 1$ (resp., $\leqslant \exp(-\alpha/2\pi)$--hence, also $\leqslant 1$), where α is any positive lower bound on the total area enclosed by the internal boundary curves of D.

P5. Let f map the interior of D into the open unit disc, with $f(0) = 0$, and suppose that some(free)component of ∂D is mapped onto part of the unit circle. If, in addition, f maps D conformally onto D_2' (resp., D_3'), then $|f'(0)|$ is minimal (resp., maximal).

Lastly, consider mappings onto the 'annular domains', D_1' and D_6'. Slightly more generally, let $\mathfrak{D}$ be bounded externally by the unit circle, and, internally, by a concentric circle of radius r, $0<r<1$, and by a finite number of continua between these two circles. Let f map the unit circle onto itself, and the circle of radius r onto a concentric circle of radius ρ.

P6. If f maps $\mathfrak{D}$ conformally onto D_1' (resp., D_6') then $\rho \geqslant re^{J}$--hence, also $\geqslant r$ (resp., $\leqslant re^{-J}$--hence, also $\leqslant r$), for $J := B/2\pi$, where B is a positive lower bound on the total area enclosed by the 'continua'(assuming that at least one is a closed curve).

As a parting shot, let it be remarked that, any domain bounded by k continua may be mapped conformally onto the exterior of k circles(see, e.g., Bergman(1950)).

15.8. Branching analysis for nonlinear operator equations.

The behaviour of solutions of nonlinear operator equations involving 'external parameters' can be extremely complicated.Not only is it far from trivial to find representations for the solutions;it happens,also,that particular solutions'split', or branch(or,bifurcate—a term with varying technical definitions)into two or more separate solutions,as the parameters pass through certain values.The basic problem of the field is to identify all of these sets of 'critical values',to characterize the type of branching behaviour associated with each critical set, and to develop effective methods for the(approximate)representation of the corresponding solutions.This,in its most general interpretation,is a vast undertaking, covering many mathematical fields,and requiring a wide range of techniques.Although the subject was founded,essentially,by Lyapunov(1906),and Schmidt(1908), its rigorous development is of much more recent origin,the main thrust coming from Soviet groups centered on Vainberg,and on Krasnosel'ski (broadly,emphasizing 'series expansions',and explicit operator calculations over Banach spaces),and from fundawork by Hale and several co-workers in America(where concepts of singularity theory are basic,and the structure of the 'zero-sets' of differentiable mappings is monitored,by means of certain,effectively constructible functionals).Questions of (perturbational)stability of the resulting 'bifurcation diagrams'(showing how the solutions branch,as functions of the parameters)have been studied in depth by many people(see,e.g.,Golubitsky and Schaeffer(1979),for a full treatment,and for additional references).Practical aspects of bifurcation theory are considered in a symposium edited by Rabinowitz(1977).A large bibliography of items bearing on the singularity-theoretic aspects of bifurcation theory may be found in Poston and Stewart(1978).There is much scope for symbolic computation in both of the main approaches to branching theory.In particular,the papers by Chow,Hale and Mallet-Paret(1975a,b),offer immediate possibilities for constructive schemes(see, also,Hale,in Knopps(1977)).However,in the present subsection,attention is focussed on the other approach,following the methods of Krasnosel'ski et al.(1972),and of Vainberg and Trenogin(1974),which are especially apt for symbolic analysis.Only the barest outline is attempted here;I hope to study some of these matters in future papers.

The method to be outlined here combines the original Lyapunov/Schmidt reduction of the branching problem from an infinite-dimensional one to a finite-dimensional one,with systematic use of (an extension of)the 'Newton polygon construction'(see Section 15.2,also)for the determination of the fractional power series corresponding to the branches of algebraic and algebroidal functions.It is claimed(see Chow,et al.(1975a),that this method cannot deal completely with braching phenomena in some cases where there are several parameters,rather thanjust one--this being one motivating factor in their work.However,the situation seems not to be clear-cut,since Vainberg and Trenogin(1974) state(para.24)that such extensions _are_ possible.A more serious problem arises from the essentially _local_ for of the solutions produced by series methods,algorithms for'global continuation'being difficult to formulate,though properly chosen transformations to new parameters may offer help in this direction(see,e.g.,Rosenblat(1979)).The principal aim of this subsection is to sketch the forms of several symbolic computation routines required to cover applications of the series method to a wide variety of problems;detailed implementations of these routines would be quite feasible,using some of the packages described earlier in this paper,though they present a formidable challenge to designers and users alike.

The Lyapunov/Schmidt and Newton polygon methods may be combined to provide a powerful,general procedure for determining the 'small solutions'(i.e.,those solutions that are continuous,and vanish at the'origin')associated with the following types of problems:systems of _implicit functions_; systems of _nonlinear integral_,and _integro-differential equations_ of the Lyapunov/Schmidt type; integral equations of general type; singular integral equations; _periodic solutions of nonlinear_(ordinary)_differential equations_; problems in _perturbation theory_; and,nonlinear equations involving _operators beween Banach_ (sub)_spaces_. All of these classes of problems are treated,formally,by Vainberg and Trenogin(1974),using a uniform method.The resulting computations are,however,so vast and unwieldy that only with the intelligent use of symbolic computation can they be accomplished at all,in nontrivial cases. Thus,this area furnishes an excellent testing ground for the power and versatility of symbolic analysis systems.When a single parameter is involved,all of the above classes of problems may be subsumed in the the following,general problem:

Find all small solutions of the equation $\Sigma H_{ik}x^{i}\lambda^{k} = \underline{0}$,as(fractional)power series, say,$x_j(\lambda)$,the sum being over all nonnegative integers,i,k,such that $i + k \geqslant 1$.It is convenient to separate the terms F_{01},and $F_{10} := -B$,and to write $F_{ik}/i!k!$,for H_{ik}. In these terms,B is a linear operator between Banach(or,topological)spaces E and E_1,$F_{01} \varepsilon E_1$, λ is a scalar parameter,and the F_{ik} for $i + k \geqslant 2$ are the(Fréchet) partial derivatives of order i in x and k in λ,at $x = \underline{0}$, $\lambda = 0$;this generalized Taylor series is studied in Hille and Phillips(1957),and Lyusternik and Sobolev (19),for instance.

Let the basic equation, $Bx = F_{01}\lambda + \Sigma F_{ik} x^i \lambda^k$ $(i + k \geq 2)$, be denoted by (*). The main problem for the solutions of (*) may be reduced (by means of an extension of the Lyapunov/Schmidt procedure) to that of finding all small solutions of the corresponding (system of) <u>branching equation</u>(s), which have the form:

$$(**) \qquad \Sigma_1 L^{(i)}_{(k)_r 0}(\xi^k)_r + \Sigma_2 L^{(i)}_{(j)_r k}(\xi^j)_r \lambda^k = 0 ,$$

where $(p)_r$ denotes the multi-index, $p_1 \ldots p_r$, $(\eta^q)_r$ stands for the product $\eta_1^{q_1} \ldots \eta_r^{q_r}$, Σ_1 has $k_1 + \ldots + k_r \geq 2$, and Σ_2 has $j_1 + \ldots + j_r \geq 0$. All realisations of (**) will be subject to these conventions, which will not be repeated. It is for (**) that the methods based on Newton's polygon are developed. The connection between (*) and (**) is, explicitly, as follows: the multiple 'power series', defining a function of λ by $w(\lambda) := \Sigma_3 \Upsilon_{(h)_r}(\xi^h)_r \lambda^k$ (where Σ_3 is subject to the condition $h_1 + \ldots + h_r + k \geq 0$, and the ξ_s are (fractional power series) solutions in λ determined from (**), continuous, and vanishing for $\lambda = 0$) gives all small solutions of (*), provided that all combinations, $(\xi_1^{(i)}, \ldots, \xi_r^{(i)})$ of small solutions of (**) are used in forming $w(\lambda)$, and the superscript, (i), takes the values, $1, \ldots, r$, in turn--so that (**) is, in general, a (finite) <u>system</u> of equations.

The plan now is to outline the symbolic computational programs which (in principle) could implement the compound procedure just described, in various contexts. First, it is necessary to explain the Newton polygon method, as it is used here. Let f be a function of two complex variables, locally expressible as a double power series in these variables--about any chosen point, (a,b): $f(z,\zeta) = \Sigma A_{\alpha\beta}(z-a)^{\alpha}(\zeta-b)^{\beta}$, the minimal value of β being ≥ 2. Then, the standard implicit function theorems give no information about the solvability of the relation $f(z,\zeta) = 0$, for either of the 'variables' in terms of the other. Instead, a method originally due to Newton (1969(!)), having a distinctly geometrical flavour, may be used to derive all possible expansions for the required solutions. If all of the pairs of exponents, α, β are plotted as points in the plane, then a polygon is determined by the <u>convex-down hull</u> of the points (α, β), relative to the origin. Thus, this is, in general, a polygonal line, rather than a closed polygon; but, together with the two axes, the hull does define a closed polygon, provided that only the 'descending' part of the hull is taken. As it turns out, only this part of the hull is relevant for the determination of small solutions (and it has always only finitely many vertices). By convention, if several of the points are colinear, the maximal possible number of them is taken to define the side of the polygon in question.

Let (α_1, β_1) and (α_2, β_2) be the end points of a segment of the Newton polygon, (α, β) denoting extra points on this segment,and (α', β'),all other points corresponding to the series expansion of $f(z\zeta)$.By expressing the expansion as a sum of three parts,corresponding to this system of labelling,writing the equation of the segment containing (α, β) as $r\alpha + s\beta = p$,and setting(if $\beta_1 > \beta_2$) $A_{\alpha_1\beta_1}u^{\beta_1-\beta_2} + \Sigma A_{\alpha\beta}u^{\beta-\beta_2} + A_{\alpha_2\beta_2} =: \chi(u)$,one may show that the equation defining the implicit relations,namely,$f(z\zeta) = 0$,takes the form $t^p\psi(t,u) = 0$, where $\psi(t,u) := \chi(u)u^{\beta_2} + t\pi(t,u)$,where $\pi(t,u)$ is a polynomial in t,u.If $\varkappa$ is any simple zero of χ,then the usual form of implicit function theorem implies that u has a unique expansion of the form $u(t) = \varkappa + et + \dots$ (assuming that the substituions $z = a + t^r$, $\zeta = b + ut^s$,have been made to obtain the form $\psi(t,u)$). In this event,one has, $\zeta = b + \varkappa t^s + et^{s+1} + \dots$ .If,however, $\varkappa$ is a multiple zero of χ,then the whole procedure must be re-applied to ψ(taken to be expanded about the point $(0, \varkappa)$—this process being repeated until a simple zero is encountered.(See Bliss(1966) for this version of the construction,together with a proof that it covers all possible solutions).Vainberg and Trenogin analyse the polygon procedure in considerable detail,in order to identify special cases for use in applications.In what follows, various schemes,S_k,are given as outlines on which symbolic computational algorithms could be based.Taken together,these schemes cover the full range of applications of the central method.The potential algorithms are either treated directly(albeit,formally)in Vainberg's work,or else, they may be constructed,using fairly straightforward procedures—the more urgent matter being their efficient implementation in a symbolic computation system,which must await future experiments.

S_1:Some preliminary reductions.

This,and the next,scheme refer mainly to the determination of sets of implicit functions from systems of nonlinear equations.When the implicit functional relationships are expressed in terms of functions analytic in all of their arguments, the corresponding branching systems may be put into the form

$$(") \qquad \Phi_i(\xi_1,\dots,\xi_m;x_1,\dots,x_s) = 0 \text{ ,for } i = 1,\dots,n \text{ ,}$$

where m,s and n are determined by the conditions of the problem,the Φ_i are analytic in all variables near the origin,and vanish there.Although this is a somewhat specialized version of the general type of branching system,it turns out that many of the apparently more complicated cases may,eventually be reduced to this form.Moreover,the case $s = 1$ occurs more frequently than the rest.For ("), the basic problem is to determine the ξ_j as continuous functions of the x_k , vanishing at the origin.When $s = 1$,it is convenient to put $x_1 =: \lambda$.

For the application of Newton's method,it is useful to transform the Φ_i into certain special forms.First:the system (") is said to be <u>regular relative to</u> ξ_j, if (for n = 1),the terms independent of λ may be put into the form $c_{\sigma j}\xi_j^{\sigma}j$ + ... + $Q(\xi_1,\dots,\xi_m)$,where Q contains no power of ξ_j alone,and $c_{\sigma j} \neq 0$. A function of the form $G(\xi_1,\dots,\xi_m,\lambda)$ is a <u>pseudopolynomial</u> in ξ_j,if it may be written as $\xi_j^q + H_{q-1}\xi_j^{q-1} + \dots + H_1\xi_j + H_0$,where $q \geqslant 1$, and all of the H_r are analytic at the origin,in the variables ξ_h ,for h,for $h \neq j$.If,further,all of the H_r vanish at the origin,then the(pseudo)polynomial is called <u>distinguished</u> (of degree q) in ξ_j--any choice of j from the numbers 1,...,n being allowed. It may be shown that there exists a nonsingular,linear transformation of the vector, $\underline{\xi}$, taking the system (") into regular form relative to <u>all</u> of the ξ_j ;and this transformation is constructible,in many cases.Thus,the <u>first step</u> in scheme S_1 is: <u>To construct(if possible)a 'regularizing transformation' for the given system,(")</u>.

Next,two functions,F and f,are called <u>equivalent</u> ,if F = fg,where each of F,f, vanishes at the origin,but g does <u>not</u> vanish there.Clearly,the systems of functions f_i and F_i,i = 1,...,n,are equivalent for small solutions,provided that f_i is equivalent to F_i,for each i,in the sense just defined.If a regular branching system is transformed into an equivalent form,in which each function is a distinguished (pseudo)polynomial (relative to the same variable,say, ξ_j),then the transformed system is said to be in <u>normal form</u> (relative to ξ_j).A constructive version of the Weierstrass Preparation Theorem(see,also Section 15.2)may be used to effect this transformation,and this is the <u>second step</u> in scheme S_1.Several variants of theseprocedures are possible,and the case where $n > 1$(so that all of the ξ_j must be determined as functions of n variables,instead of just λ) may be treated algorithmically,in certain circumstances,though the critical case is n = 1,for general application,and,in what follows,it is assumed that algorithms to effect the necessary transformations have been implemented.

S_2: Evaluation of the resultant in the case: $m = 2 = n$.

This a special case, but it is of interest for potential symbolic computation routines. One starts from the system of equations $\Phi_i(\xi_1, \xi_2, \lambda) = 0$, for $i = 1,2$, where the Φ_i are analytic near $\underline{0}$, and ord $\Phi_i(\xi_1, \xi_2, 0) \geq 2$ (i.e., that $\Phi(\xi_1, \xi_2, \lambda) = O(\lambda^2)$, as $\lambda \to 0$, the 'O-constant' depending on the other variables). Assume that the system is reduced to normal form $G_i(z_1, z_2, \lambda) = 0$, for $i = 1,2$, where the G_i are distinguished polynomials, and form the resultant--say, $\Psi(z_2, \lambda)$--of the G_i, relative to z_1, over the ring, $K[[z_2, \lambda]]$, which is a unique factorization domain. As is well known, this resultant may be represented as a determinant, whose entries are the 'coefficients' from the 'polynomials' G_i, together with zeros, all in prescribed locations (see, e.g., Van der Waerden(1949)). (Since several of the packages discussed in this paper have a 'resultant facility', this step should not cause undue problems--except, as usual, in simplification. Moreover, it may be necessary to introduce approximations at some stages of this evaluation; but, this is in line with the philosophy adopted here, provided that adequate estimates are available for the errors incurred). In order to produce tractable output, it may be possible to represent $\Psi(z_2, \lambda)$ as a (double) truncated power series; for, only small solutions are required, and the relevant properties of Newton's polygon, as applied to Ψ, are fixed by the terms up to some finite order (determined by ord $\Psi(z_2, 0)$, and ord $\Psi(0, \lambda)$. Thus, the aim of S_2 is to produce output of the form: $\Psi(z_2, \lambda) \triangleq \Sigma p_{kh} z_2^k \lambda^h$, for $k = 0, \ldots, K$ and $h = 0, \ldots, H$, say. In the applications, the highest (nonnegative) power of λ that can be removed as a factor, say, λ^σ, is used to define a 'reduced' resultant: $\lambda^\sigma R^*(z_2, \lambda) := \Psi(z_2, \lambda)$--so that, the approximations introduced to write $\Psi(z_2, \lambda)$ as a truncated power series produce corresponding approximations for $R^*(z_2, \lambda)$.

The importance of R^* in the analysis of solutions of the original operator equation (which takes the form of a pair of functional equations in the present case--call them (+)) is as follows. (a) The system (+) has only finitely many small solutions IFF $R^*(z_2, \lambda)$ does not vanish identically. (b) If $R^*(0,0) = 0$, then the components, $z_1(\lambda), z_2(\lambda)$, of all small solutions of (+) have convergent (fractional) power series representations of the forms:

$$z_1(\lambda) := \Sigma\, b_{\nu j}^{(\mu)} \lambda^{j/r_\nu}, \text{ where } z_2^{(\mu)}(\lambda) := \Sigma\, a_{\mu i} \lambda^i \text{ is the standard}$$

power series representation for $z_2(\lambda)$, the r_ν are integers, and $1 \leq \nu \leq h_\mu$, say, for $\mu = 1, \ldots, L$, and each of i, j, goes from 1 to ∞. (c) If $R^*(z_2, \lambda)$ vanishes identically, then (+) has one or more families of small solutions (together with a finite collection of small solutions, associated with the splitting of the Φ_i into nontrivial factors--which the vanishing of the resultant guarantees).

The role of scheme S_2 is to test whether $R^*(z_2,\lambda)$ vanishes identically(by finding a suitable representation where every term is demonstrably 0;or else,by verifying that this occurs for all terms up to some prescribed order--though this is not conclusive);or,alternatively,to identify a pair z_2', λ,for which $R^*(z_2', \lambda')$ is nonzero.Again,the value of $R^*(0,0)$ may be found,and,if this is 0,and condition (a) holds,then one knows that there are only finitely many small solutions of (+), and that each has the representation given in (c)--from which,in some cases,the unknown coefficients may be found,recursively(in conjunction with the information obtained from consideration of Newton's polygon:see scheme S_3).When approximate solutions can be obtained from (c),they must be cast into forms suitable for use in other stages of the calculations.Since several packages have good facilities for handling truncated power series,full advantage must be taken of this mode of representation.

S_3: Special cases of Newton's polygon.

The main task of this scheme is to test for(and then apply) certain conditions on the coefficients of the (truncated)power series associated with the given operator equations--and with entities derived from them as a result of various types of approximation procedures.These conditions are analysed,briefly,but systematically, by Vainberg and Trenogin,and it is a straightforward matter to incorporate expanded versions of their results into a general scheme,in such a way that the implications,for numbers and types of small solutions,are readily available.It should be possible to include additional examples whenever this seems desirable, so that the potential range of application of this scheme is maximized.Further, it should be noted that,when the given conditions involve coefficients of the branching system other than the very early ones,the corresponding number of possible configurations for Newton's polygon becomes large;so it is necessary to construct routines for deriving the properties of solutions associated with each configuration.This,too,may be accomplished(at least,up to a fairly high order of coefficients);though much experiment will be required to arrive at satisfactory routines covering nontrivial cases.As mentioned before,even the 'low-order configurations',although apparently simple,have important implications for the solutions of sophisticated problems(e.g.,those involving integro-differential equations),since such problems give rise to approximations for which the basic configurations determine the behaviour--with regard to the number,and type of solutions. Consequently,it is highly desirable to have efficient implementations of the basic routines corresponding to conditions on the low-order coefficients.This, then,is the primary objective of scheme S_3

S_4: Formation of branching systems for sets of implicit functions.

In S_3, it was remarked that the application of 'special cases of Newton's polygon' gives vital information about the number and form of small solutions, in all types of operator equations—but, especially, for operators corresponding to the solution of a set of implicit functional relations. However, the problem of actually obtaining the branching system must not be overlooked, as it is one of considerable complexity, in all but fairly trivial cases. Indeed, most of the other schemes to be mentioned in this subsection involve, principally, the formation of the branching (usually, in some approximate representation) for various classes of operator equations. In the present case, the general form of the branching system—if the functions determining the implicit relations are analytic in a neighbourhood of the origin—is: $\Phi_i(\xi_1,\dots,\xi_m;x_1,\dots,x_s) = 0$, for $i = 1,\dots,n$; and the case of greatest application has $n = 1, x_1 := \lambda$. However, for the purposes of a symbolic computational scheme, the general case should be covered (the aim being to obtain, ultimately, a collection of truncated Taylor series representations involving all of the above variables—to prescribed orders); after which the deductions based on special cases of Newton's polygon (scheme S_3) may be made—along with attempts to determine sufficiently many of the unknown coefficients associated with the expansions given in scheme S_2(c). Ultimately, truncated fractional power series are obtained for the various possible solutions indicated by the results of applying S_3 in this case. All of this may be accommodated naturally within a symbolic computation package.

In what follows, an idea is given of how the branching systems for various types of nonlinear operators may be obtained in approximate forms suitable for symbolic analysis. Details are kept to a minimum, but a certain amount of notation is inescapable. There are, always, two separate stages: first, the formal specification of the branching system, with unknown coefficients, and second, the (approximate) determination, using recursive procedures, of these coefficients. Only after this is it possible to analyse the resulting branching system (using, e.g., other schemes, S_j) in order to discover the number and nature of its small solutions—and hence, of the small solutions of the original operator equations. In all cases, the problems encountered are very suitable for symbolic analysis, and the treatment offered by Vainberg and Trenogin, while totally impractical without symbolic computation, is yet excellently fitted for development into approximation routines. The notions of Krasnosel'ski et al. (1972)—especially, those revolving around the idea of approximate branching systems—may be used, too, in arriving at satisfactory versions of these routines.

S_5: Branching systems for (sets of) integro-differential equations.

As for all of the more general classes of equations, the aim is to bring the branching system into a canonical form, so that its small solutions may be identified, and then determined, approximately, using Newton's polygon method. A crucial part is played by the linear part of the nonlinear operator, through the distribution and muliplicities of its eigenvalues. For linear operators, the bifurcation points coincide with the eigenvalues; and, for certain classes ofnonlinear operators, it may be shown that every point of bifurcation is also an eigenvalue. A technique introduced originally for dealing with linear integral equations, may be used to modify a linear operator, A, to $\tilde{A}$, in such a way that a given eigenvalue of A is not an eigenvalue of $\tilde{A}$ ('Schmidt's Lemma'). In the case of integral equations, where Af means $\int Kf$, for some kernel, K, an eigen value, μ, of order N, with corresponding, linearly independent eigenfunctions, $\phi_1, \ldots, \phi_N$, orthonormal over the given domain, and 'associated eigenfunctions', $\psi_1, \ldots, \psi_N$, of the the adjoint operator, A^*, for the same eigenvalue, μ, one replaces K by $\tilde{K}$, where $\tilde{K}(u,v) := K(u,v) - \mu \sum \psi_i(u) \overline{\phi_i(v)}$. For general operators, an analogous procedure is followed. This representation of the modified operators is basic in the formation of their branching systems, as the following sketch indicates.

The equation to be studied may be written in the form:

$$(*) \qquad (I - A)u = \Gamma_{01}(s;v) + \Sigma\, \Gamma_{mn}(s;u,v) \ ,$$

where I is the identity operator, $Au := \int Ku$, $\Gamma_{01}(s;v) := K_0(s)v(s) + \int K_1(s,t)v(t)dt$, and $\Gamma_{mn}(s;u,v) := \sum_{\nu} \int \cdots \int K^{(\nu)}(s,t_1,\ldots,t_r)u(s)^a v(s)^b \Pi u(t_j)^{a_j} v(t_j)^{b_j}\, dt_1 \ldots dt_r$,

all integrations being over a basic domain, say J, the product going from 1 to r, and the sum over ν from 1 to some value N_r. If, in addition, the conditions $a + a_1 + \ldots + a_r = m$, $b + b_1 + \ldots + b_r = n$, hold, then $\Gamma(s;u,v)$ is called an integral power form of order m, in u, and n, in v. The summation in equation $(*)$ is taken over all m,n such that $m + n \geq 2$; and this constitutes an integral power series. The extension of these ideas to the case of several functional arguments is direct; and substitutions of such forms and series into each other may be effected in the obvious way--subject to convergence requirements. The kernels, $K_0, K_1, K^{(\nu)}$, are assumed to be continuous, real-valued or complex-valued functions over appropriate cartesian products of J with itself.

With these preliminaries, the method may be summarised briefly. If 1 is <u>not</u> an eigenvalue of the operator A in (*), then, it may be shown that (*) has a unique solution, continuous over J, expressible in the form $U(s) = \Sigma V_i(s;v)$, where $V_1 = \Gamma_{01}$, the other V_j are integral power forms in v, and it is assumed that the condition $|V_1(s;v)| \leqslant \delta$ holds, for a suitably small number, δ. Suppose, now, that such a solution has been analysed, and justified rigorously; and that an equation of the form $(*)_N$ is given, for which 1 is an eigenvalue of order N of the operator A. <u>There is no problem in constructing well-regulated approximations to the unique solution when 1 is not an eigenvalue of A</u>, and <u>this is the first task to be included in scheme</u> S_5. It is complicated, but poses no special problems of principle, so it will not be discussed further here, beyond saying that a recursive scheme may be given for determining the V_i, and that this scheme may itself be adapted for effective symbolic computation. Let the kernel in $(*)_N$ be modified (using Schmidt's procedure) so that 1 is <u>not</u> an eigenvalue of the modified kernel, say, $\tilde{K}$, derived from K, and denote the corresponding modification of $(*)_N$ by $(*)_0^{\sim}$, and the eigenfunctions associated with 1 for $(*)_N$, by ϕ_h, ψ_h (as before). Next, define parameters ξ_j, by $\xi_j := \int u(t)\overline{\phi(t)}dt$, where u is now the unique solution of $(*)_0^{\sim}$ —which depends, in turn, on all of the ξ_q, and may be obtained (approximately) by treating $(*)_0^{\sim}$ as linear equation in u (i.e., ignoring the occurrence of u on the 'RHS', to a first approximation), introducing the corresponding Fredholm resolvent, and using this to represent the solution, say U, as a function of s, with 'parameters' ξ_q. By substituting U into the defining relations for ~~the~~ ξ_q, (and by putting $v(s) =: \lambda v_0(s)$, where v_0 is any suitably chosen, 'fixed' function) one derives 'consistency conditions' of the form:

$$\xi_j(s) = \int \{ \Sigma' \xi_j \overline{\phi_j(s)} + \lambda g(s) + \Sigma'' a_{n_1 \ldots n_p k}(s) \lambda^k \Pi \xi_i^{n_i} \} \overline{\phi_j(t)} dt \quad (**) ,$$

where Σ' is from 1 to p, and Σ'', for $n_1 + \ldots + n_p \geqslant 2$. The equation from which U is derived by successive approximation may be written as:

$$u(s) = \Sigma \xi_j \phi_j(s) + \lambda g(s) + \sum_{m+n \geqslant 2} \lambda^n \int_J g_{mn}(s,t) u^m(t) dt ,$$

after the introduction of λ (so that only the power forms in u are involved, effectively).

The conditions (**) may be rearranged to assume the form of the general branching system;in particular,all of the'branching coefficients',$L^{(i)}_{n_1\ldots n_p0}$,and $L^{(i)}_{n_1\ldots n_pk}$, may be identified.Accordingly,the basic steps in approximating the branching system in this case are as follows.(i)Reduce the givenkernel in the 'linear part of the equation' to a kernel for which 1 is not an eigenvalue.This may involve several types of approximation,but the techniques are familiar,and should not present serious problems for symbolic computation.Call the new(approximate) kernel so obtained K*.(ii)Form the resolvent of K*.This also,requires the use of various approximation procedures--the final result being a truncated double power series.Again,no special problems should arise,though the determination of error estimates is essential).Moreover,the g_{mn} may not be determinable exactly, and,in any event,only finitely many of them can be used--which introduces certain approximations even befor the resolvent of K* is formed;but,all of this is OK, too, provided that adequate error estimates are obtained.(iii) Use the approximate form of (**) in the defining relations for the ξ_j.This is a matter of substitution and simplification,making high demands on any system handling it.The 'coefficient functions',$a_{n_1\ldots n_pk}(s)$,may be found recursively,and the approximations involved here may include those required to evaluate the integrals,as well as others inherent in the recursion procedure itself.Even if these integrals can be evaluated exactly,e.g.,using a variant of the Risch algorithm,the results may be too complicated for further calculation to be practicable.Consequently,some kind of approximate integration may be desirable--either by using a quadratue procedure,in 'symbolic form',or else,by approximating the integrand,and then performing the integration exactly(see,e.g.,Ng(1979)for some comments on this idea). (iv)Denote the approximately determined branching system by (***).This will have the form of a truncated multiple power series in the ξ_j and λ,for which the coefficients are the 'approximate branching coefficients'.It may be seen that the Newton polygon method(used in conjunction with scheme S_3) yields information as to the number and type of the small solutions of the original equation.Thus,the next step is:apply S_3.

In the interests of simplicity,all of these steps have been summarised very briefly.However,the determination of the (approximate)branching coefficients,although straightforward mathematically,constitutes a major computational problem,for which very large resources are required.Basically,one must determine the functions $a_{n_1,\ldots,n_pk}$,recursively,before the branching coefficients can be found.The corresponding sets of functional recurrence relations are given in some detail by Vainberg and Trenogin,but they are not solved--even partially.It is here that symbolic computation is indispensable.The possibility of recursive determination of the $a_{n_1,\ldots,n_p}$ is assured because each'coefficient' depends,in a suitable way, only on those with 8lower sets of subscripts'.All of this may be adapted for symbolic analysis,fairly directly.

(v)The next stage of the solution procedure consists of substituting formal(fractional)power series in λ —one collection for each set of values of the ξ_j — into the original equation(s),in order to find the unknown coefficients in these fractional power series.It is here,above all,that the information derived from Newton's polygon,used on the branching system,is crucial,since it guarantees that a solvable system of recurrence relations is obtained.Without this prior information,it is by no means clear that any set of relations obtained by uninformed, 'sausage-machine tactics' would be solvable at all;and it might be necessary to do a great deal of computation before this unsolvability manifested itself.Thus, in spite of the fact that the mere determination of (approximate) branching systems involves much calculation,without this preliminary activity,no rigorous mode of solution is possible.It follows that,the essential task,for all sophisticated symbolic computation in this area(and,indeed,more generally) is to find sufficient conditions for the compound approximations used to provide adequate information for the meaningful characterization of small solutions of operator equations.

The extension of these methods to certain types of integro-differential equations is comparatively direct.If only the first derivative,Du(s):= du(s)/ds is involved, then the introduction of extra integral power forms,say $\nabla(s;u,Du,v)$,allows the previous calculations to be imitated to the point where(approximate) branching systems are derived.In principle,by using terms of the form $\Delta(s;u,Du,\ldots,D^pu,v)$, a formal theory analogous to the one already given may be obtained;but there are analytical difficulties hidden in this prescription,which must be investigated in each case separately.Again,the original equation may be '$D^ju(s) = \ldots$ ',and this, too,may be included in the scheme outlined here.Many variants of the procedures just described could be given.In particular,Krasnosel'skii et al.(1972) offer a more succint presentation(though,an equivalent one),and(as mentioned above),a formulation of 'asymptotic approximations of branching systems',which will be valuable in the general investigation of branching phenomena for nonlinear equations.In the remaining 'schemes' of this subsection,a much briefer sketch is given of applications to other types of equations.

S_6:Branching systems for periodic solutions of(sets of) differential equations.

Only a very brief outline is given here.The basic equation is of the form:

$$\frac{dx}{dt} = f(t,x) + \lambda g(t,x,\lambda) \qquad (*)$$

where each of f,g and x has n components,and f,g are jointly continuous in all variables,and ω-periodic in t;while, f is holomorphic in x,and g,in x and λ-- so that,each has a (multiple) power series representation.It is assumed that (*) has an ω-periodic solution,say ϕ,for $\lambda = 0$;and the main problem is:to find ω-periodic solutions, $\psi(t,\lambda)$,such that $\psi(t,0) = \phi(t)$.The branching systems for this problem are derived from Poincaré's solution of the initial value problem corresponding to: $x(0,\alpha,\lambda) = \phi(0) + \alpha$,in the form

$$x(t,\alpha,\lambda) = \phi(t) + \chi(t,\underline{\alpha},\lambda) \qquad (**) ,$$

with χ holomorphic in the components, $\alpha_1,\ldots,\alpha_n$,of $\underline{\alpha}$--and,hence,representable as a (truncated) power series,whose coefficients may be found by equating the coefficients of 'like' monomials in $\alpha_1,\ldots,\alpha_n,\lambda$.This produces a recursive system of differential equations for the χ-coefficients,$c^{(i)}_{k_1\ldots k_n k}$,with the initial conditions : $c^{(i)}_{k_1\ldots,k_n k}(0) = 1$,if $k_i = 1$ and $k_j = 0$, for $j \neq i, k = 0$;and $=0$,otherwise.Since the solution (**) is periodic in ω IFF χ is,it follows that (**) is ω-periodic IFF each component of χ satifies an initial condition,namely:

$$\chi_i(\omega,\alpha,\lambda) - \phi_i(0) - \alpha_i = 0 \qquad (***).$$

If the maximal nonnegative power of λ is divided out of (***),then the resulting system,say,(***)',has the form $\Psi_i(\alpha_1,\ldots,\alpha_n,\lambda) = 0$,where $i = 1,\ldots,n$,and the functions Ψ_i are holomorphic at the origin.The following result may be used now.

Theorem: the small solutions of (*) and (***)' are in one-to-one correspondence.

In order to derive a branching system,let M denote the Jacobian matrix of the Ψ_j relative to the α_k--evaluated at the origin,and let r be the nullity of M,so that, n-r unknowns may be eliminated,nontrivially,from the conditions (***)', to produce, after redefining,and,possibly,relabelling,the variables,the 'reduced system'

$$\Phi_j(\xi_1,\ldots,\xi_r,\lambda) = 0 \qquad (***)'' ,$$

for $j = 1,\ldots,r$,which,if put in terms of a (truncated) multiple Taylor series, has the canonical form for a branching system.

Thus, scheme S_6 must include routines to perform the following tasks (with adequate error estimates at each stage). (i) Given (*) and (**), determine (approximately) the functions Ψ_i. (ii) Construct the matrix, M. (iii) Find the nullity of M. (iv) Perform the elimination converting (***)' into (***)". (v) Convert (***)" into a form suitable for the application of the other schemes, S_k.

This sort of treatment may be extended to : (non)autonomous systems; quasi-linear systems; equations having singular solutions (in the sense that $\overline{\lim}_{\lambda \to 0} \| x(t, \lambda) \| = \infty$); and, solution of equations involving Banach spaces (see, also, scheme S_7). Some questions of stability of solutions may be considered, too. All of these procedures are amenable to interpretation in forms suitable for symbolic analysis.

S_7: Branching systems for operators between Banach spaces.

Here, there are two cases to consider: those of Fredholm operators, and of (singular) Noether operators. In both cases, certain finite-dimensional subspaces play a basic part, and the use of bases in these spaces makes it possible to derive branching systems analogous to those found in less abstract settings. Thus, a full, approximate tretment is feasible. This general framework may be shown to subsume many special problems of practical interest, including: singular (nonlinear) integral equations with 'Cauchy' or 'Hilbert' kernels; second-order elliptic boundary-value problems with parameters, subject to 'Dirichlet', Neumann', or 'directional derivative' boundary conditions; and, certain Nth-order elliptic boundary-value problems. The symbolic routines from Section 15.6 (calculus in Banach spaces) may be used both in this work and in S_8 (perturbation calculations), as well as in several other types of applications dealt with in this paper. Indeed, this 'cross-application' is one of the prime aims of symbolic analysis. For instance, in Section 15. , some possible routines involving bases in Banach spaces are considered; and the potential use of these routines in studying nonlinear operators between Banach spaces, and in finding 'best' approximations for functions into such spaces, is of great interest. Moreover, each of the special problems just listed has an associated branching system, which may be put into a form sufficiently concrete for computation. The details are somewhat technical, and are not given here; but the definitions of Fredholm and Noether operators are worth giving, since they include definitions of the associated finite-dimensional subspaces, on which the computational treatment is founded, for S_7 and S_8.

Let γ be any element of the Banach space of bounded linear operator between the given Banach spaces E_1 and E_2, and denote by $N(\gamma)$ the kernel of γ (i.e., the inverse image of the zero element in E_1). It may be seen that the dimension of $N(\gamma)$ equals the number of its linearly independent elements. The operator γ is called normally solvable if either (a): the equation $\gamma x = h$ is solvable for any h in E_2; or else, (b): $\exists$ $N^*(\gamma) \subset E_2^*$ (and $\not\subset \{0\}$) such that '$\gamma x = h$' is solvable IFF $\forall \psi \in N^*(\gamma)$ $<\psi, h> = 0$, where the angular brackets stand for 'the value of ψ at h', and E_2^* denotes the space adjoint to E_2. In these terms, γ is a Fredholm operator if $\dim N(\gamma) = \dim N^*(\gamma) = n$, say, where n is finite. In this event, each of $N(\gamma)$, $N^*(\gamma)$ has a basis consisting of n elements. If, instead, the dimensions of $N(\gamma)$, $N^*(\gamma)$ are, respectively, m and n—finite but distinct integers—then γ is called a Noether operator.

As is well known, many of the classical results involving implicit functions may be extented to 'calculus in Banach spaces, and branching theory is concerned with monitoring the solutions of equations of the form $F(x,y) = \underline{0}$, for suitably analytic functions, $F: E_1 \times E \rightarrow E_2$, where all of E_1, E and E_2 are Banach spaces. In this setting, Frechet derivatives are prominent, and the most basic result is a natural extension of Taylor's theorem. There are, also, 'implicit operator theorems', which guarantee the existence of a unique solution, say, $x(y)$, under specified restrictions (analogous to the classical ones). In view of the part played by the finite-dimensional spaces N and N^*, the routines in S_7 must cover aspects of both linear algebra and analysis. As usual, the primary aim is to obtain reliable approximations to the branching systems, in various cases, in representations allowing the application of routines from other schemes, S_j; which means that the representations must be as concrete as possible, with readily applicable error estimates.

S_8: Branching systems associated with problems in perturbation theory.

The principal problems of interest here are of two types.(a)To study the perturbations of a linear operator by a'small (non)linear term';and (b):to investigate the branching behaviour of eigenvalues and eigenelements of Fredholm operators. Both of these topics may be developed constructively with the help of Jordan chains of (Fredholm) operators,whose essential properties are as follows. Let γ, with associated spaces N , N* (see S_7) be a Fredholm operator,and $\phi = \phi^{(1)} \varepsilon$ N. (Unless it is stated explicitly,N and N* always refer to γ,here).Let a finite sequence of elements $\phi^{(j)}$ be defined by the conditions:

(*) $\quad \gamma\phi^{(1)} = \underline{0} \;\; ; \;\; \gamma\phi^{(k)} = A\phi^{(k-1)}$,k = 2,...,p ,where $<A\phi^{(p)}, \psi_i> \neq 0$.

Here,the ψ_i span N*,and the $\phi^{(j)}$ are called A-associates of ϕ (of order j). When N(and hence,also,N*)is one-dimensional,the A-associates may be made unique, and one writes $J(\phi, \gamma, A) = p$,and calls p the length of the A-Jordan chain for the element ϕ relative to γ,the uniqueness being achieved by imposing the extra conditions $<\phi^{(j)}, \mu> = 0$,for j = 2,...,p,and any fixed μ in E_1^*.More generally, if dim N $\geqslant 2$,then the set of all A-associates(of any order)of basis elements of N,is called an A-Jordan set(relative to γ)—such a set being complete,if $\det(\alpha_{ij}) = 0$,where $\alpha_{ij} := <A\phi_i^{(p_i)}, \psi_j>$,where the $\phi_i^{(j)}$,j = 1,...p_i,are the elements of the Jordan chains for the basis elements, ϕ_i,of N.The relevance of these definitions for the present work may be illustrated by a basic theorem.Let a normally solvable,Fredholm operator, γ,be given,such that $\gamma - \sigma A$ has a bounded inverse,provided that the modulus of σ is sufficiently small.Then,all A-Jordan chains of N have finite length.Moreover,a complete A-Jordan set exists IFF the above invertibility condition on γ holds.

The relation of these results to perturbation theory is clear if one considers the equations(for y): $\gamma y = h + \sigma Ay$ (**) , γ,A map the Banach space E_1 into E_2 and γ is a Fredholm operator—all subject to the invertibility condition just given.If n:= dim N,then there are three cases:n = 0;n = 1,and,n $\geqslant 2$.For n =0, γ is invertible,and $(I - \sigma\gamma^{-1}A)^{-1}$ has an(absolutely and uniformly)convergent series expansion,form which $y(\sigma)$ may be found in the form $\Sigma y_k \sigma^k$,through the termwise evaluation of $(I - \sigma\gamma^{-1}A)^{-1}h$.This procedure could be adapted to a concrete algorithm for symbolic computation.

When $n=1$, the only branching behaviour of solutions occurs for the operator, $\tilde{\gamma}$, where $\tilde{\gamma} y = \gamma y + \langle y, \mu \rangle z$, μ being the element used to 'normalize' the A-chain (as above). If p is the length of the Jordan chain defined by the(unique)element of N(since $n=1$), and q is defined as: $q=0$, if $\langle h, \psi \rangle = 0$; and, $q = \min(s \; \langle T^s h, \psi\rangle \neq 0)$, where $T := A\gamma^{-1}$, otherwise, then the case '$p=\infty, q<\infty$' may be shown to correspond to a one-parameter family of solutions: $y(\sigma) = \tilde{\gamma}^{-1} h + \xi(\sigma)\phi$, for an arbitrary function, ξ. Here, z is an element of E_2, and the procedure may be framed in a concrete form. Lastly, if $n \geq 2$, then the analysis is more complicated, but still potentially amenable to symbolic computation. It turns out that the equation (**) does have a solution, $y(\sigma)$, and that this solution is analytic in a neighbourhood of the origin--unless certain conditions hold on the numbers, p_i, q_i, analogous to p,q for the case $n=1$; in which case, the solution is analytic only in an annular neighbourhood of the origin.

The second major problem(branching behaviour of eigenelements of Fredholm operators)gives rise to branching systems, which may be approximated in much the same way as has been indicated in more concrete situations. Let $\sigma \in C$, $|\sigma| \leq \rho$, and consider an operator $A(\sigma): E \to E$, the eigenspace corresponding to an eigenvalue, λ, having dimension $n \geq 1$. Assume that $\gamma = A - \lambda I$ is a Fredholm operator, and that $A(\sigma)$ is continuous, in the uniform operator topology, with $A(0) = A$—a given operator. <u>Basic problem</u>: find the eigenvalues, $\lambda + \mu(\sigma)$, of $A(\sigma)$, where $\mu \to 0$ with σ; and find the corresponding eigenelements. If one puts $A - A(\sigma) := H(\sigma)$, then the eigenvalue equation may be written as: $\gamma y = H(\sigma) y + \mu(\sigma) y$. By introducing the operator, $\tilde{\gamma}$, defined as $\gamma + \sum_1^n \langle \cdot, \mu_i \rangle z_i$, where, if (ϕ_i) is a basis for $N(\gamma)$, (μ_j) is orthogonal to (ϕ_i), and (z_k) is orthogonal to (ψ_i), and by putting Γ for $\tilde{\gamma}^{-1}$, one may define elements $a_{ik}(\sigma, \mu)$ as:

$$a_{ik}(\sigma,\mu) := \langle [H(\sigma) + \mu I][I - \Gamma H(\sigma) - \mu\Gamma]^{-1}\phi_i, \mu_k \rangle .$$

Then the branching system has the form : $\Lambda(\mu,\sigma) := \det(a_{ik}(\mu,\sigma)) = 0$. (+). Accordingly, the first task for scheme S_8 is to form an adequate approximation to the equations (+); after which, a method for approximating the resulting elements, $\mu_j(\sigma)$, must be found. If $A(\sigma)$ is analytic, and one postulates a representation $a_{ik}(\sigma,\mu) = \sum \alpha_{ikrs} \mu^r \sigma^s$, then (+) takes the form $\sum L_{rs} \mu^r \sigma^s = 0$, the summation being subject to $r+s \geq n$.

If Newton's polygon is used to study this branching system, then it may be shown that (i)For all small enough, positive σ, there are only finitely many eigenvalues $\lambda(\sigma):=\lambda+\mu(\sigma)$, with $\mu(0)=0$;(and there may be no such eigenvalue).(ii)Each eigenvalue of this type corresponds to finitely many eigenelements, $y(\sigma)$.(iii)Every eigenvalue, or eigenelement, has a convergent series expansion in(integral or fractional)powers of σ.The various possible cases are determined by properties of Jordan chains of I-associates of basis elements, ϕ_i, of $N(\gamma)$.

When $n=1$, one has $y(\sigma)=[I-\Gamma H(\sigma)-\mu\Gamma]^{-1}\phi$, where μ is to be replaced, successively, by the $\mu_j(\sigma)$. If $p=1$(e.g., if γ is a hermitian operator on a Hilbert space)then $\lambda(\sigma)$, reducing to λ at the origin, is determined uniquely as an eigenvalue of $A(\sigma)$, for 'small σ'; and each of $\lambda(\sigma)$, $y(\sigma)$, is analytic. Next, if $p>1$, finite, then, with proper multiplicities, there exist exactly p eigenvalues reducing to λ for $\sigma=0$, for $A(\sigma)$. One such eigenvalue has an expansion in integral powers of σ; while, the remaining p-1 eigenvalues have fractional power series expansions, all based on the same fractional power of σ--namely, on $\sigma^{1/p}$. In fact, on the assumption that $A(\sigma)=A-\Sigma H_i\sigma^i$ (summed for $i\geq1$), and that $L_{01}=$ $=\langle H_1\phi,\psi\rangle\neq0$, it may be proved that:

$$\mu_k(\sigma)=[-L_{01}\sigma]_k^{1/p}+o(\sigma^{1/p});$$

$$y_k(\sigma)=\phi+[-L_{01}\sigma]_k^{1/p}\Gamma\phi+o(\sigma^{1/p}),$$

where $[\quad]_k^{1/p}$ denotes the k th value of the p th root(e.g., $e^{k(2\pi i/p)}w$, for w^k). All of these procedures may be elaborated into algorithmic forms, for which symbolic computation is feasible.

Lastly, in this subsection, the scope of computational procedures will be summarised, so that the objectives of anygeneral 'branching package will be clear. The sequence of operations is as follows. Formation of the branching system; qualitative analysis of the branching system; computation of series solutions(in several stages); description of local behaviour of solutions near branching points; enlargement of the domain of definition of solutions(this could lead to another scheme, say, S_9, where conditions for(the absence of)secondary bifurcations, and related matters, would be covered, along with methods of continuation--see, e.g., Rosenblat(1979)for some examples relevant to hydrodynamics; also, Pimbley(1969), for detailed results on certain types of integral equations); and, symbolic/numerical interface(routines for performing the numerical components of calculations for which all of the basic analysis is done symbolically). Although there are many aspects of branching phenomena only barely mentioned here(and, some omitted altogether)--e.g., the methods based on singularity theory(including catastrophe theory), their potential suitability for intelligently directed symbolic analysis should be evident to anyone studying them. I hope to investigate specific algorithms in these areas eventually.

15.9 .Function-theoretic methods in elasticity.

In this subsection, the methods of interest are all related to the use of contour integrals—either in conjunction with conformal mapping, or else, with 'potential distributions', whose associated 'densities' over the boundary are to be determined. Only one method will be treated in any detail, but this gives a good indication of the general possibilities. Naturally, very similar methods may be used for other types of problems than the elasticity boundary-value problems for which most of these techniques were developed. The basic references for this work are Sokolnikoff(1956), Muskhelishvili(1963), Green and Zerna(1968), Timoshenko and Goodier(1970), and England(1971). These works include many references to the original papers. All of this work refers to two-dimensional problems; but some constructive('potential theoretic' methods have been given by Kupradze(1965), and they offer scope for symbolic computation. Another useful reference for function theoretic methods is Mikhlin(1957), where several types of problems are considered. The main difficulties arise from the adequate determination of the mapping functions—especially, for general, n-connected domains, and in the possible extension of these ideas to cover anisotropic materials. In this connection, it is clear that the procedures developed by Lekhnitskii(1963), involving the use of functions of several complex variables, offer excellent possibilities for symbolic analysis, and, as such problems have not been solved effectively up to now, even for general 1-connected domains, the effort involved in constructing procedures covering this class of problems will be worthwhile. As far as detailed algorithms for the classical problems are concerned, Kantorovitch and Krylov(1958) contains interesting material. One other idea stems from Muskhelishvili's result that, if the unit disc is mapped conformally onto a given domain by means of a rational function, then, the fundamental boundary-value problems of elastostatics may be solved exactly, in terms of Elementary functions for that domain. Since, however, quite general types of functions may be approximated(sufficiently uniformly) by sequences of rational functions(see, e.g., Smirnov and Lebedev(1968)) it may be conjectured that the corresponding solutions tend(in some sense) to that for the general domain. This idea is very suitable for exploration using symbolic computation. Remarks on the possible use of symbolic computing for problems in plane elasticity may be fou nd in an unpublished report by the author(Elvey(1978))— on which the rest of the material in this subsection is based. This deals with one method(due to Sherman(1940)) for solving the fundamental boundary-value problems of plane elastostatics, for isotropic bodies. Although the method is so old, and many calculations have been done, the possibility of its effective implementation within some symbolic computation package does not seem to have been considered. Since the method covers n-connected domains, its approximate implementation would be valuable. As usual, many intermediate approximations would be required to realise the complete algorithm, but this aspect of the problem is not stressed here. Again, it may be feasible, eventually, to include anisotropic materials; but this, too is ignored, here.

The equations due to Sherman refer to the boundary value problems in which (a) the stresses are prescribed on the boundary(of the given,n-connected domain), and (b) the displacements are given on the boundary.A basic idea in the application of complex analysis to problems of elasticity is the representability of a <u>biharmonic</u> function in terms of a pair of <u>analytic</u> functions;and it is these analytic functions which are,in turn represented as Cauchy integrals,for which the 'boundary density functions' are to be determined.In fact,a single density suffices in each case,the form of the corresponding analytic functions being known(by'analogy'), and the (generally distinct)constants associated with disjoint components of the boundary being expressed in terms of this same density function--call it μ .If the total boundary be denoted by L,then the basic equation may be written as:

$$(*)\qquad \varkappa\,\mu(\tau) + (\varkappa/2\pi i)\int \mu(t)\,d\log \alpha(t,\tau) - (1/2\pi i)\int \overline{\mu(t)}\,d\,\alpha(t,\tau) = \chi(\tau)\ ,$$

subject to the representations of the functions,say,f and g as explicit expressions involving the density function(there is no need to give them here).It will be indicated how an effective reduction of this problem to a succession of approximation procedures may be accomplished,provided only that the boundary curves are known in suitable parametric forms,the curve L_0 containing n-1 curves, L_j.The steps in this procedure may be summarised as follows.

(A)Representation of the boundary curves in parametric form.(B)Specification of the kernels appearing in (*).(C)Formal representation of the 'boundary constants' in terms of the unknown density function,using contour integral forms,and making use of the parametrizations of the curves.Bearing in mind that there are n boundary curves ,and that τ is regarded as 'fixed',while t varies on these curves(e.g., for purposes of integration),one may obtain,eventually,a pair of Fredholm integral equations(discrete variants of the ones give,e.g.,by Muskhelishvili(1963)), which constitute a system of 2n equations for the values of the real and imaginary parts--say u,v,of the density function on each of the curves L_j.This system has the form (step D):

$$(**)_1\qquad W_j(h_0) + \pi^{-1}\,\Sigma\int\{2W_k \sin^2\theta_{jk} - W_{n+k}\sin 2\,\theta_{jk}\}\theta'_{jk}\,dh + \ldots$$

$$+ \int_{L_j} W_j(h)T_j(h)\,\theta'_j(h)\,dh + \Sigma\,F^0_k = A^0_j$$

$(**)_2$ Interchange j and n+j,k and n+k on all <u>singly</u>-subscripted variables in $(**)_1$.

If one defines elements Y_{rs}, E_r, for $r,s = 0,\dots,2n-1$, by:

$$Y_{jk}(h,h_0) = 2\pi^{-1}\sin^2\theta_{jk}\,\theta'_{jk} + \delta_{kj}T_k\,;\quad Y_{j,n+k} = -\pi^{-1}\sin 2\theta_{jk}\,\theta'_{jk} + \delta_{kj}T_k\,;$$

$$Y_{n+j,k} = -\pi^{-1}\sin 2\theta_{jk}\qquad ;\quad Y_{n+j,n+k} = 2\pi^{-1}\cos^2\theta_{jk}\,\theta'_{jk}\,;$$

$$E_j = A_j^0 - \Sigma F_k^0\qquad ;\quad E_{n+j} = A_{n+j}^0 - \Sigma F_{n+k}^0\,,$$

(where the dependence of all of these functions on h,h_0 has been suppressed), then the following 'canonical system' of integral equations is obtained (step E):

$$(***)\qquad W_r(h_0) + \Sigma\int Y_{rs}(h,h_0)W_s(h)dh = E_r(h_0)$$

where r varies from 1 to 2n-1, s is summed over the same range, and the integration is over the basic parameter interval, say from a to b.

If, now, the <u>displacements</u> (rather than the stresses) are given on the boundary, then a slightly different equation is derived for the density function. After similar 'discretization' procedures(again, using the parametrization of the boundary), one gets another pair of Fredholm integral equations for the real and imaginary parts of the density function—in terms of which the pair of analytic functions is determined, for the representation of the biharmonic function solving the original problem. Once again, if cognizance is taken of the fact that there are n-1 boundary curves, and if suitable changes of variables are introduced, then, a canonical form, analogous to (***), say, (***)', may be derived. A comparable treatment for 'mixed' boundary-value problems is possible, too(see, e.g., Mikhlin(1957)), though there are some extra complications in this case.

The potential use of symbolic computation in this kind of calculation(as well as in the calculations based on the prior determination of suitable conformal maps), should be evident. All of the procedures involved are quite capable of adaptation so that approximations can be made, error estimate incorporated, and approximate symbolic representations of the relevant stress and displacement functions can be determined. The aim of this subsection is merely to demonstrate the feasibility of such an enterprise, and to show how an approximation may be obtained in terms of the solutions to standard systems of Fredholm integral equations—for which many different types of procedures have been developed. All of the remaining steps in any of these procedures involve various specific types of approximations in the evaluation of the integrals, etc., which occur; but the optimal choice of these depends(often, strorgly) on the particular analytical properties of the functions to be approximated, of the integrands encountered, and so on.

15.10. Approximate solution of ill-posed problems.

Until fairly recently, problems modelled by differential (or other types of) equations for which the solutions were not continuous functions of the data--relative to specified topologies--were discarded as inappropriate or meaningless (in much the same way as complex numerical solutions of equations were ignored in an earlier era). Of course, this situation arose, principally, from the observations of Hadamard(1902,1932), who introduced the concept of 'well-posedness' in order to delineate those types of boundary data, etc., which were acceptable in the formulation of problems modelling physical processes. No doubt, there has been an over-reaction against problems which are not well-posed in Hadamard's sense. This is most unfortunate, since, it may be shown that 'ill-posed' problems span a substantial body of (pure and applied) mathematics, so that it is essential to devise methods for 'solving' them in some manner. This subject is dealt with elegantly by Tikhonov and Arsenin(1977), who demonstrate that there are several ways in which the concept of 'solution' can be modified, in order to restore the property of 'stability against small changes in the data'. In the following sketch, the sources of ill-posedness in a number of basic problems is indicated--after which some remarks are made on the possible use of symbolic computation in forming approximate (generalized) solutions.

(a) Solution of $Af := \int Kf = u$, for a function f with given u. Let $u^* := u + A\psi$. Then, one has, $\| u^* - u \|_{L^2} = \{ \int |A\psi|^2 \}^{1/2}$. If, for instance, $\psi(t) := N \sin \omega t$, then $(A\psi)(t) = o(1)$ as $\omega \to \infty$ (Riemann-lebesgue Lemma), whereas, $\| f^* - f \|_{C(J)} = |N|$ can be made arbitrarily large, by suitable choice of N, where $C(J)$ is the space of functions continuous on the interval, J, over which the integrals are taken. Moreover, it is easy to see that $\| f^* - f \|_{L^2} = O(N)$; so the instability persists in this case, also. Again, it may happen that u is known exactly, but that A is such that criteria for the existence of solutions of '$Af = u$' cannot be applied. In this event, since the existence of a classical solution cannot be guaranteed, a quasi-solution, $\hat{f}$, is defined by the requirement that $\rho_U(A\hat{f}, u)$ be minimal, ρ_U being some metric on U, the set of all 'given functions', u.

(b) Differentiation of partially-known functions. Since the problem of finding the n th derivative of a function, u, may be reduced to solving (for f) the integral equation: $\int_0^t (t-\tau)^{n-1} f(\tau)\, d\tau = (n-1)!\, u(t)$, as may be verified by induction on n, the ill-posedness here follows from that covered in case (a).

(c) Numerical summation of Fourier series. Here, it is simply a matter of allowing the (Fourier) coefficients to be 'perturbed' by (arbitrarily small) terms, whose sum diverges for some value of the argument of the function being represented as a Fourier series.

(d)The Cauchy problem in the plane,for $u(x,y)$,with $u(x,0) = g(x)$,and $u_y(x,0)= h(x)$, is unstable under'small changes' in g and h.(This is Hadamard's original example;. for the proof of instability,note that, for the Laplace equation,with $g(x) \equiv 0$, and $h(x) := n^{-1}\sin nx$,the solution is $u(x,y) = n^{-2}\sin nx \sinh ny$,in '$y \geqslant 0$';whereas, the solution is $u(x,y) \equiv 0$,if $h(x) \equiv 0$).

(e)Analytic continuation from an arc into a domain. Let f be given on an arc,S, within a domain,D,where the distance from S to ∂D is d.Then,it is easy to see that function elements differing(in the 'sup norm')arbitrarily little on S,may produce analytic continuations whose difference has unbounded sup norm over D.

(f)For the so-called inverse gravimetry problem,it may be shown,on certain assumptions,that the 'boundary curve' separating materials of different densities (under the earth's surface)may be determined from the 'gravitational anomolies produced at the surface,through the nonlinear integral equation $Bf = v$,where $(Bf)(x) = \int \log \{ [(x-\xi)^2 + b^2] / [(x-\xi)^2 + (b - f(\xi))^2] \} d\xi$;and this equation may be shown to be unstable against perturbations in v.

(g)Among many other problems giving rise to instabilities,are:the solution of singular systems of linear algebraic equations(because the precise evaluation of a numerical determinant is ruled out,in general,by 'round-off,and other errors— so that,arbitrarily small changes in the coefficients can change the nature of the system .When equations with symbolic coefficients are given,this difficulty cannot occur;but the verification that a determinant is(or,is not) zero,may demand storage beyond the computer's capacity);certain(linear)programming problems ,with imprecisely specified data;and more general optimization problems for functionals(where the sequence of elements producing a minimal value for the functional need not,itself,converge to a solution of the problem for which the functional is defined).

These,and many more problems are discussed in detail by Tikhonov and Arsenin (1977).Even though the instabilities are manifested,generally,in numerical calculations,the possibility of forming meaningful symbolic approximations to the 'solutions' of ill-posed problems is of great interest—the definition of 'solution' being so framed that stability is re-established.The proposed methods of constructing generalized solutions include the techniques of:'quasi-solution', 'selection',replacement',and 'quasi-inversion'(all used when the set of'possible solutions' is compact);and the method of 'regularization',for problems where the potential solution-set is not compact.The dominant modes of approximation amount to minimization of suitably-defined,smooth functionals—and prior restriction of the domains on which operators act.These procedures(almost all of them due to various Soviet mathematicians)are formulated in a functional-analytic manner,very apposite for the development of symbolic-analytic algorithms.The possession of symbolic approximations to generalized solutions,which exhibit desirable stability properties not shared by the classical solutions,could prove just as valuable, in certain cases,as for the solutions of well-posed problems.Purely numerical solutions cannot encompass the same range of analytical behaviour as can their symbolic counterparts;nor can they reflect adequately the instabilities inherent in the original problems(especially,when 'extra parameters' are present).In short, there appears to be considerable scope for symbolic analysis in this area--though great care must be taken to ensure that the irregularities in behaviour are not lost in oversimplified approximate representations.For applications of these, and related,concepts,to questions involving partial differential equations,see, e.g.,Knops (1972) .In general,the class of 'inverse problems'(typically,the determination of various operators from functionals of solutions of the corresponding operator equations--e.g.,to find the coefficients in a differential operator)tend to be ill-posed;but they include many important applications;so,once again,the need for systematic methods of solutions is clear.A good treatment of this subject(mainly,in relation to ill-posed problems for partial differential equations in mathematical physics)is given in Colton(1976a,b),where several constructive procedures(involving 'Bergman integral operators')may be found--many of them suitable for effective implementation in symbolic analysis.

15.11. Design of switching circuits, using Galois fields.

The main reason for including this topic here, is that some of the methods adduced (especially, by Romanian researchers) for the synthesis of complex switching circuits —e.g., for the control of complex traffic systems—constitute direct applications of 'function-theories in prime-power Galois fields (including, even, analogues of the Lagrange interpolation formula!). As such, they offer excellent opportunities for productive symbolic computation. More generally, it is evident that many problems in the constructive analysis of (finite) automata are amenable to symbolic computational treatment (see, for instance, Kobrinskii and Trakhtenbrot (19), where several potential algorithms may be found; and, also, Wang (197), for other examples). The comparatively simple problems associated with switching circuits are emphasized in this subsection because of their accessibility to symbolic computation; but there is no doubt that analogous treatments can be developed to cover a wide range of problems in this area.

A very full exposition of the 'Galois field method' is given in Moisil (1969), where the computations (many of them carried out in detail), have, already, the appearance of output from well-tuned symbolic analysis routines. It is certain that similar calculations (but, to much higher orders) can be done automatically; and, that variants involving other types of algebraic structures can be explored—possibly, having applications in the design of the intricate micro-chips upon which future generations of symbolic computation systems will depend (especially, since a complete 'time-evolutionary description' can be given, for each component of a 'network', which is of fundamental importance in some models of parallel computation—see Section 12.7). These remarks are not intended to denigrate the efforts of the many people who have brought this theory of circuit synthesis to such a sharply algorithmic state (introducing a variety of subtle concepts, in the process). It is regrettable, however, that so much of the repetetive work had to be done in full, when a few, properly-designed computation routines (one for each class of circuits) could have produced so much more—and, allowed the 'interactive development of optimal designs'. This reaction is familiar in relation to any area of research where symbolic analysis has been (or, can be) successful; the early symbolic computing applications in relativity superceded, in a few seconds, calculations that required the ingenuity and perseverence of several people, over a period of months—even years, in some cases. Thus, for switching circuits, it would seem that the algebraic theory, though attractive as a construction, is suitable only for computer implementation—even if this was not envisaged when the theory was developed!

In this theory,a typical problem has the form: determine the characteristics of a circuit consisting of'contacts',$c_1,\ldots,c_L$,where c_t has s_t possible 'settings', and the mode of interconnection(i.e.,'series',or 'parallel')of the contacts is prescribed.A k-position contact is associated with the cartesian product of the Galois fields G_{p^i},where the numbers p^i are the prime-power factors of k;and,a collection of contacts having different numbers of possible settings may be handled by including in the cartesian product all relevant prime-power factors. In many important applications,however,the structures are comparatively simple, and the results are obtainable in explicit forms.Of particular interest is the occurrence of various algebras corresponding to many-valued logics(see,also, Section 14,for some comments on this).The variety of networks that can be covered by this theory is considerable,and includes:networks with k-position contacts;with polarised relays;with 'slowly-acting'relays,and,combinations of these types.With each network,a 'working function' is associated.The construction of working functions for specified circuits involves combinations of(generalized)Boolean and field operations.The most basic question,for any network,is:When does a current flow through the circuit?(i.e.,for which arrangements of settings of the component contacts does the working function take the(Boolean)value 1?).In this sense, the analysis of a circuit is accomplished when its working function has been determined in an accessible form—and,this would be one major task for symbolic analysis.Other fundamental problems include the reduction of networks of multi-position con contacts to 'equivalent' networks of two-position contacts;and,the synthesis of networks exhibiting prescribed operating characteristics.(Plainly,not every set of proposed operating conditions can be realized—many problems are 'ill-posed'.Conditions for 'well-posedness',in this sense,may be derived by using known compatibility criteria;and this aspect,too,may be covered in symbolic computation routines).Apart from its obvious importance in practical applications,the algebraic theory governing circuit synthesis offers many attractive opportunities for studying,in a practical setting,certain parts of number theory,field theory, and formal logic;it provides,also,an excellent grounding for the systematic attack on design problems for more sophisticated automata of deterministic type.With suitable modifications,some of these techniques could be used in the study of automata having some stochastic elements,too.Lastly,it is notable that circuit synthesis may be studied(predominantly)by any of the following methods(apart from the Galois field procedures):'Boolean algebras';'many-valued logics',and 'graph theory'.Some of these methods are used in the algebraic approach,but only peripherally.There may be scope for symbolic analysis in the other approaches, too,though it seems unlikely that they would so ideally suited to symbolic computation as the'Galois method'.

15.12. Stability analysis for(systems of) (non)linear differential equations.

Here,again,the invitation for systematic symbolic analysis is an open one,as,the the field has reached a point where many types of behaviour of solutions(e.g., asymptotic boundedness and stability)may be analysed on the basis of various crieria involving function(al)s of the coefficients—and certain procedures are framed explicitly in terms of approximations.Although a vast literature exists on these matters,a most convenient starting point is proovided by Sansone et al. (1974)(which includes many references to the original papers),where a diverse collection of constructive procedures is given—often in forms easily adaptable for symbolic analysis.The possibility arises,also,of solving 'inverse problems', in which a(system of)nonlinear equation(s) is constructed,whose solutions exhibit specified asymptotic behaviour,or stability characteristics.There are also varied applications to problems in control theory.Indeed,constructive treatments of asymptotic and stability analysis for automatic-control systems are given by Zubov(1962),including a detailed study of Lyapunov functions(with some emphasis on effective methods);transient processes in (non)linear systems;construction of solutions(with estimates of the influence of perturbations on their behaviour); and,studies of almost periodic oscillations in nonlinear systems.All of these procedures are excellently fitted for investigation using symbolic analysis.There are,also,problems associated with the design of networks of nonlinear electrical devices(transistors,etc.),which lead to systems of differential equations,for the analysis of which some of the above-mentioned techniques may be useful(though, variants of'implicit function theorems' play a dominant rôle here:see,e.g., Willson(1974)).

In another direction,the analysis may be pursued of the time-evolution generated by (quite simple) nonlinear transformations(e.g.,those associated with various 'oscillators'),having practical applications in such crucial fields as the study of vibrations in certain buckled structures,and the stability of floating 'oil platforms'.Here,the theory of'almost periodic systems'is relevant.However,the potential value of symbolic analysis in this area lies(principally)in the fact that some of these 'oscillator transformations' are either known to possess (or else,suspected of possessing) so-called strange attractors(subsets of the phase space having a very intricate structure,and producing apparently chaotic behaviour in the solutions of purely deterministic problems)whose (non)existence depends ultra-sensitively on the'initial conditions'imposed.(See,e.g.,Holmes(1979), where the equation $\ddot{x} + a\dot{x} - bx + cx^3 = \lambda \cos \omega t$ is studied in great detail;and,Ruelle(1980),for a recent expository paper).The 'fine structure' of the strange attractors cannot be elucidated adequately by numerical computation alone(indeed,certain constructive procedures arise naturally here—see,for instance, Melnikov(1963)).A second,comparatively elementary source of possible strange attractors is furnished by certain nonlinear algebraic transformations,which,when repeatedly iterated,produce phase portraits associated with strange attractors. It is notable that an extensive study of iterated nonlinear transformations in two,or three variables was undertaken long ago,by Stein and Ulam(1964);and that the first identification of a strange attractor seems to be that of Lorenz(1963), in a hydrodynamical context.However,Stein and Ulam do not pursue this idea.

One further area where symbolic computation could prove to be of great value is the systematic study of of bifurcation phenomena describable by variants of the Hopf bifurcation.See,e.g.,Marsden and McCracken(1976),for constructive procedures to test whether the Hopf bifurcation theorem applies in any given situation—and whether,if it does apply,the resulting periodic orbits are stable.The algorithm for testing stability is given in considerable detail,and it appears to be implementable within some of the systems discussed in this paper.As usual,the gain accruing to the use of symbolic computation is twofold:analytical interrelations may be examined reliably,and whole families of cases may be studied at a stroke—by varying suitably defined parameters.The highly constructive nature of many of the criteria pertaining to the study of asymptotic states of dynamical systems(which covers all of the topics mentioned so far,as well as several others, related to the solutions of more general operator equations)suggests that symbolic analysis could be most valuable here.There are ,also,other aspects of the qualitative theory of operator equations than the ones emphasized in this subsection; in particular,the investigation of equations in which the 'coefficients' are almost periodic functions raises many fascinating problems,and requires new techniques(see,e.g.,Fink(1974)where an extensive list of references is given—along with several potentially effective procedures).Since most 'apparently periodic' phenomena encountered in applications are in fact(at best)almost periodic,it is important that the analysis for this class of equations be developed as constructively as possible.If the resulting methods are combined with the general procedures(of which Lyapunov's method is the most familiar),and with techniques designed specifically for handling bifurcation problems(see,also,Section 15.8) then a package of wide applicability will be obtained,which,in view of its ultimate aims,must be provided with direct access to a high-level numerical language.

15.13. Basic calculations in(algebraic)topology.

The main aim of this subsection is to mention some procedures in topology which are prohibitively complicated for 'hand calculation'(in all but essentially trivial cases),but which are of great importance in almost all constructive topological investigations.It is highly probable that the GROUP system(see Section 11),and some of the procedures listed in the computer-based bibliography maintained by Velsch(1978-)at the University of Aachen,would allow many types of calculations other than the ones suggested here to be handled in symbolic analysis;but the proper consideration of these matters must be undertaken by experts in (pure and applied)topology.Here,only one type of application is cited: the use of topological methods in the analysis of electrical networks.There is a large literature even in this single area,but an idea of the sort of clacula-tions that may be attempted may be gleaned from Kim and Chien(1962),and from the remarkable set of papers known collectively as the RAAG Memoirs(see Kondo(1955, 1958,1962,1968)) in which comparatively sophisticated topological techniques are used to analyse networks containing several species of components.One other application of topological methods(to the location of fixed points of mappings)is treated,briefly,in Section 15.16.

Three types of calculation are considered here :(i)identification of triangulable spaces,and determination of appropriate simplicial decompositions of the corresponding polyhedra(as well as,'assembly' of polyhedra from collections of simplexes);(ii)effective implementation of the constructions embodied in the simplicial approximation theorem(in both its 'weak' and 'strong' forms),for use in other procedures;(iii)calculation of some groups of basic importance in certain non-trivial cases(including the 'fundamental group',and the (singular)homology groups).

Notes.(a)The 'simplicial fixed point algorithms' correspond to hypotheses weaker than those required for the various'contraction-mapping principles'(which incorporate simple error estimates).(b)The essential task(once the simplicial approximation theorem has been used)may be reduced,often,to the reconstruction of a group from its set of generators and relations--which can be implemented for symbolic computation,in many cases.The most general reconstruction problem may be only incompletely soluble:the existence of such a group is guaranteed;but there may be no effective procedure for deciding whether it has any element distinct from the identity.Moreover,it may not be possible to decide(finitely) whether two given 'words' of a group,G,are transformable into each other by inner automorphisms of G;and this has implications for the design of simplification procedures in symbolic computation(see,also,Section 13).However,it is unlikely that these extreme cases would be encountered in routine topological calculations;so,it should be possible to give an effective reconstruction scheme.

(i)Triangulation and simplicial decomposition.
A (geometric)simplex is just the direct,k-dimensional analogue of a tetrahedron (for k a nonnegative integer).A (geometric) simplicial complex ,K,is any collection of simplexes(all contained in some Euclidean space R^m) such that (α) each face of any simplex in K is also in K,and (β) the intersection of any two simplexes from K is itself in K.The dimension of K is the maximum of the dimensions of its'member simplexes'.Any subset of K for which property (α) holds,is called a subcomplex(say,L) of K.A simplicial n-ple consists of an(ordered)collection,say, $(K,L_1,\dots,L_{n-1})$,where the L_j are subcomplexes of K.If K is given the induced topology of R^m,then K,regarded as a point set,is a topological space(say, $|K|$)— the so-called polyhedron of K (in which case,the L_j are associated with subpolyhedra, $|L_j|$,of K).For any simplicial complex,H,denote by V_H the set of vertices in H (a typical k-simplex being determined by k+1 vertices).Then,a simplicial map , f: $|K| \to |L|$,satisfies the conditions:(s1) $f(V_K) \subset V_L$;(s2)If $\sigma \varepsilon K$,then $f(V_\sigma)$ spans a simplex in L(possibly,with repeats);(s3) f acts linearly(relative to the vertices of any simplex).A simplicial map,g,'from K to M',subject to the extra condition,g($|L|$) $\subset |N|$,is called a simplicial map of(simplicial)pairs; that is,g: $(|K|,|L|) \to (|M|,|N|)$.

In these terms,certain basic results may be stated concisely;but ,it is useful to introduce a few more definitions in order to allow a complete statement of the simplicial approximation theorem.If X is any topological space,then a triangulation of X, is any pair,K,h,such that K is a simplicial complex,and h: $|K| \to X$ is a homeomorphism.(This allows one to treat'like polyhedra' spaces which(geometrically)are not polyhedra).It may be proven that:every simplicial map between simplicial complexes is continuous.The simplicial approximation theorem states(roughly) that every continuous mapping between two polyhedra is approximable by simplicial mappings(this will be made more precise in a moment).Thus:every simplicial mapping is continuous,and every continuous mapping is 'almost simplicial' (the mappings being beween polyhedra).For general calculations,where it is inconvenient to embed all complexes in some R^k,an abstract simplicial complex,say,K*,may be defined to be any finite set of elements(the vertices of K*) together with a collection of subsets, σ^*_j,of K*(the simplexes of K*) such that:(i)any subset of a simplex is itself a simplex;and,(ii)the dimension of a simplex is one less than the number of vertices in it—the dimension of K* being the maximum of the dimensions of its simplexes.It may be shown that,all realizations of an abstract simplicial complex are(simplicially)homeomorphic;and that,every abstract n-dimensional simplicial complex has a realization in R^{2n+1}.In general,a given geometric simplicial complex may have several 'abstractions'.

The 'assembly' of polyhedra, generation of simplicial decompositions and 'manipulation of simplicial objects' are all tasks for which symbolic analysis can be valuable. In order to understand the significance of strong simplical approximation, one must specify procedures for the systematic decomposition of a given complex into 'arbitrarily small pieces' (such decompositions are known as subdivisions). Before this is done, however, it is useful to define what is meant by: 'g is a simplicial approximation to f'. The basic sense of approximation here is that of homotopy. Let K,L be given simplicial complexes, and $f: |K| \to |L|$ a continuous mapping. If σ is any simplex in K, then, the star of σ (in K) is the union of the interiors of all simplexes in K of which σ is a face (including, as a special case, the star of a single vertex, regarded as a degenerate simplex, etc.). Then: $g: |K| \to |L|$ is a simplicial approximation to f IFF $(\forall v \in V_K)\ f(st_K(v)) \subset st_L(g(v))$, where $st_K(u)$ denotes the star of u in K. It may be shown that, the composition of two simplicial approximations is also a simplicial approximation; and (more importantly) that every simplicial approximation of f is homotopic to f.

Subdivisions, K', of a simplicial complex, K, are obtained from the decomposition of the simplexes of K (most commonly, by decomposing each simplex, σ, of K, into the subsimplexes formed when the barycenter of σ is joined, by segments of straight lines, to the vertices of σ). This procedure may be iterated, to determine the r th subdivision, $K^{(r)}$, of K, for $r = 2,3,\ldots$. If L is a subcomplex of K, then, the derived complex, $(K,L)'$, of K relative to L (i.e., leaving L 'fixed') may be defined, inductively (on n) for the complexes $(K^n \cup L)'$, where K^n denotes the n-skeleton of K (i.e., the set of all simplexes of dimension at most n, in K). Thus, derived complexes of the form $(K,L)^{(q)}$, involve subdivisions $K^{(q)}$ leaving L fixed. The details of this procedure are somewhat intricate; but it may be made effective, and adapted for use in symbolic analysis. Next, a star covering of K is the collection of stars of its vertices (each regarded as a 0-dimensional simplex); and the mesh of such a covering is the supremum of the diameters of its stars. It is easy to see that: mesh$(K^{(r)})$ tends to 0, as r tends to ∞. It is possible, now, to state two versions of the main result.

Weak simplicial approximation theorem. Let K,L, be simplicial complexes, and let $f: |K| \to |L|$ be continuous; then $(\exists r) |\ f: |K^{(r)}| \to |L|$ has a simplicial approximation.

Strong simplicial approximation theorem. Let K,L, be simplicial complexes, and let $f: |K| \to |L|$ be continuous. Suppose that M is a subcomplex of K such that the restriction of f to $|M|$ is a simplicial mapping. Then:

$$(\exists r)(\exists g: |(K,M)^{(r)}| \to |L|)\ |\ g_M = f_M \wedge g \approx f \text{ rel } M \qquad (*).$$

In (*), h_M stands for the restriction of h to M, and '$\approx$' denotes homotopy (the 'relative' qualification meaning that the 'homotopy function', F, satisfies: $F(m,t) = f(m)$ and $F(m,t) = g(m)$, for all m in M, t in I, where I is the 'parameter interval').

The strong simplicial approximation theorem provides a powerful means of developing effective procedures for the construction of various groups associated with topological spaces.Indeed,using this theorem,it is possible to determine a finite set of generators and relations for the fundamental group of any domain homotopy-euqivalent to a polyhedron.The procedures are complicated,but,undoubtedly within the scope of properly-directed symbolic computation.The major task,for the application of symbolic analysis in this(and related)area(s),is to introduce computer representations of various topological objects in an optimal way for their manipulation within the system--and for the production of intelligible output.For instance,one routine would accept,as input,the analytical specification of a given domain,and try to determine for it triangulations(if it is not known to be a polyhedron),associated simplicial decompositions,and(repeated)subdivisions of the corresponding simplicial complexes.Another general routine would constitute an effective implementation of the procedures underlying the strong simplical approximation theorem--so that these procedures may be used to obtain(eventually)a finite set of generators and relations for the fundamental group of the domain.All of the intervening stages would have to be covered by additional routines.It is not possible to discuss these here,but they raise no problems of principle,as far as symbolic analysis is concerned.A comparable situation obtains with regard to the calculation of(singular)homology groups(for spaces homotopy-equivalent to polyhedra) with the help of'exact sequences of chain complexes'(and,in certain cases, of 'simplicial homology methods').It appears that the techniques required in these approaches,too,may be framed in effective forms,so that some calculations may be made amenable to symbolic analysis.More generally,it is clear that many of the basic procedures of homological algebra offer potential opportunities for the use of symbolic computation(see,e.g.,MacLane(1963) for a conspectus of these procedures).

Of course,this whole undertaking requires the development of highly nontrivial routines,of a kind hardly envisaged up to now;yet,in principle,there would seem to be no insuperable obstacles.As mentioned earlier,the computer-based bibliography maintained by Velsch(1978-)contains many examples of implementations of sophisticated algorithms,and the GROUP system(see Sectio 11) is well-developed.If various algorithms from these sources were adapted for use in topological procedures, and transcribed into th language of a powerful,general-purpose,symbolic computation system,then a collection of routines could be assembled,covering several types of basic calculations.As a 'test case',some of the computations could be performed for 'triangulable 2-manifolds;and it may be possible to include some less simple schemes for the classification of manifolds.The strong simplicial approximation theorem is due to Zeeman(1964);and the effective methods of calculation to Tietze,Scifert,and van Kampen(for Homotopy),and to Mayer,Vietoris, Eilenberg and Steenrod(for 'exact sequence techniques').All of these methods(and many more,covering a wide variety of calculations)are expounded in Maunder(1970), which would provide an excellent starting point for this work.

The applications to network theory,also,are susceptible to direct(symbolic) computation.It is worth remarking that many of the procedures suggested in the 'RAAG papers' require the use of differential geometry for non-Riemannian spaces(especially,in relation to electrical machines,in which contacts are 'made' and 'broken' repeatedly).Computational procedures in Riemannian geometry are well covered by existing systems(because of their relevance to relativity),but there are certain characterizations(e.g.,of 'flatness'--see,Gray and van Hecke(1979)) that could be incorporated most effectively within symbolic computation schemes, and these,even for Riemannian spaces,do not seem to be covered at the moment.A basic treatment of non-Riemannian spaces is given by Eisenhart(1927)--though,his treatment is not aimed at computational procedures.Thus:routines for basic analysis in non-Riemannian spaces would form a valuable package,both for independent use,and for use in conjunction with topological procedures.

15.14. Operational calculus:generalized functions.

One of the central propositions put forward in this paper(see,especially,Section 14) is that an attempt should be made to implement constructive(or,operational) procedures from all areas of mathematics,for use in symbolic analysis.Naturally, the degree to which this objective is attainable depends strongly on the nature of the calculations involved—and,on the realizability of genuinely constructive procedures.However,even when only very partial realizations are available,it is worthwhile giving tentative schemes,which may be improved,as the capabilities of systems are enhanced.This is the situation in the case of generalized functions (and their associated operational calculi),which form the basis for this and the next subsection.

There are several approaches to the definition of 'generalized functions',but,only two of them are fundamentally distinct—namely,the methods of 'convolution quotients'(as used by Mikusinski(1959)),and, of 'linear functionals on spaces of test functions'(as introduced by Sobolev(1936),systematized and extended by Schwartz(1950–51),and treated in great detail,for theory and applications,by Gel'fand et al.(1964–68).Given the operational definitions of certain,basic generalized functions(as determined,for instance,by their effects as factors in integrands involving suitable 'test functions'),it is possible to 'catalogue' a wide range of formulae,for possible use in the course of other investigations; and ,these formulae may be incorporated in symbolic computation packages.Moreover, if the test-function framework is adopted,sequential approximations to generalized functions may be defined and manipulated.Consequently,there are,essentially, two levels at which symbolic analysis could be used in this area.

Level 1:manipulation of known formulae.This covers the solution of certain(systems of)differential equations;the routine use of specified generalized functions (especially,of the Dirac delta distribution,and its(weak)derivatives);differentiation and integration of generalized forms of complex powers and Elementary functions;manipulation of generalized Fourier transforms;formal solution of boundary-value problems,etc..Some of these schemes would involve the use of basic function-theory (e.g.,residue calculus),and of the properties of differential forms—both of which are covered by existing systems.The use of approximations to generalized functions(e.g.,as members of sequences defining these functions)is easily handled in symbolic analysis;but the interpretation of such approximations must be considered carefully,if consistent results are to be guaranteed.On the other hand,the formal relations may be implemented exactly(with provision for substitution,when they are applied in concrete representations).The admirably clear summary given by Gel'fand et al.(1964)could serve as a starting point for the development of procedures at Level 1.For the complete implementation of 'an operational calculus' for (systems of)ordinary differential equations, with various types of initial and boundary conditions,the scheme discussed in detail by Liverman(1964)seems to offer excellent possibilities.A comprehensive package, based on an effective(approximate)realization of this scheme,could be of wide practical use(since the existing symbolic computation packages for solving differential equations are not designed to deal with distributional solutions(though, some of them can handle the analytical singularities associated with applications of the Frobenius series method).Once again,many facilities from other routines would be required--for instance,in the formal solution(by 'Kramer's rule')of systems of linear 'algebraic' equations,whose 'coefficients' are linear differential operators;but this should not present undue problems.

Level 2:general,formal manipulations. Here,the emphasis is on the manipulation of formal relations corresponding to the 'sequential representation' of generalized functions--the aim being to make possible certain procedures involving 'weak derivatives'--e.g.,in relation to analysis in Sobolev spaces(with extensive applications to the study of variational problems;see,e.g.,Ciarlet(1978),Fairweather (1978),Weisel(1979),and Wendland(1979)).Among other things,one must:deal with 'trace operators'(for specifying the boundary behaviour of functions);introduce norms appropriate for the definition of Sobolev spaces of fractional and negative orders;and,above all,formalise the basic interrelations of the objects typically generated in thecourse of calculations,to a point where 'operational rules' may be established,and incorporated in the system.This is not an easy task,but (subject to sensible initial limitations)it does merit exploratory investigation--starting from the simplest possible analytical situations,and extending the procedures,gradually,to cover problems of greater complexity.
In another direction,the methods of Mikusinski(which may be used in some cases where the 'functional' methods fail)could be formulated effectively,too--since they are designed specifically for operational use.The approximations introduced in such an implementation would stem,not from imprecision in realizing the operational relations themselves,but,rather,from the spectrum of analytical problems emanating from factorization,expansion in series,indefinite integration,etc.;in other words,from the problems of approximate determination of functions,and of the results of applying analytical operations to them.In Section 15.15,a sketch is given of the method of Ehrenpreis(1970),in which the operational techniques developed for ordinary differential equations are extended(in a specified sense) to systems of partial differential equations with constant coefficients,using techniques of considerable sophistication and complexity,which,even in their simplest realizations,would require many of the other potential symbolic computation routines outlined in this paper.

15.15. Operational methods for partial differential equations.

After the remarks,in Section 15.14,on the possible implementation of operational calculi for (systems of) ordinary differential equations,it is appropriate to consider now how certain operational procedures for the (formal)solution of partial differential equations might be realized within a symbolic computation system. Two general approaches are considered here:the treatment(aimed,primarily,at solving equations with constant coefficients) due to Ehrenpreis(1970),whose central concern is the extension of the notion of 'Fourier transform' to cover'functions of exponential growth at infinity';and,the more formal,operational scheme developed by Maslov(1976),to attack diverse problems in theoretical physics,which may be reduced to the solution of partial differential equations over Hamiltonian manifolds.On the face of it,neither of these theories looks very promising from the viewpoint of symbolic analysis;however,because of their ultimately practical objectives,each of them has a core of constructive procedures whose approximate implementation would be,on its own,of potential value(especially in the construction of asymptotic solutions).The intricacies if these theories are so great that only the barest sketches can be presented here;but,since detailed accounts are given in the references cited,my objective may be limited to showing that there are opportunites for the creative use of symbolic analysis in this area.

The evolution of Ehrenpreis' theory may be summarised conveniently by examining the domains of validity of successive theories of 'Fourier integrals'(for functions of n variables).The classical theory(see,e.g.,Titchmarsh(1948))applies only to'functions of small growth at infinity'(typically,to elements of the function space L_p,for $1 \leqslant p \leqslant 2$).In the next stage,functions of'essentially-polynomial,but non-exponential,growth at infinity' were included(see,e.g.,Schwartz (1950-51)).The third stage(Ehrenpreis) allows for functions of exponential growth at infinity;but this is possible,only if the 'frequencies' appearing in the Fourier-type integral of such functions may be complex (so that convergence is assured).If this is accepted,however,then problems of nonuniqueness of representation arise,since(by 'Cauchy's integral formula')one has: $\exp\{i\zeta z'\} = (2\pi i)^{-1}\int_\Gamma (w-z')^{-1}\exp\{i\zeta w\}\,dw$ .Even so,it may be possible,in some cases,to restore the crucial uniqueness property by restricting the'permissible values' of the frequency to suitable subsets of C^n $(n \geqslant 1)$.A subset of this kind(when it exists)is called sufficient for the definition of a Fourier-type integral of a given function.A particular subset may be sufficient for a whole class of functions:indeed,several results of this kind are known.Often,the dimension of a sufficient set is found to be half of that of the basic space(i.e.,it is n,when $C^n \approx R^{2n}$ is involved);but,there are also many situations of practical importance in which the sufficient sets have dimension 0,and it is in these cases that symbolic analysis may be used most effectively.The central concept of Ehrenpreis' theory is that of 'analytically uniform spaces'(roughly,the most general framework within which Fourier-type integrals 'make sense',and can be manipulated);and the main result(the so-called fundamental principle) asserts that:if V is the subspace of an analytically uniform space,W,of distributions,defined(for a given, linear partial differential operator,P) by $V:=\{f\in W \mid P(D)f=0\}$,then the subset of C^n,say, $\delta_P:=\{\zeta \mid P(\zeta)=0\}$ is(essentially) sufficient for V (where,the meaning of 'essentially' can be made precise).The importance of this result is that,any element,T,in W,may be represented as a Fourier-type integral:

$$(*)\quad T=\int_{C^n}\omega(z)(1+|z|)^{n+1/2}d\mu(z)/k(z)\ ,$$

where μ is the Radon measure on C^n defined by $d\mu(z):=(1+|z|)^{-2n-1}\overline{F(z)}\,|dz|$.
Here,F is the 'Riesz function' generating a specified linear functional on $L^2(C^n)$,and k is a 'majorant function'(i.e.,a positive-valued,continuous function on C^n).In particular,if T is a function,or a (Schwartz)distribution,then,the integral in (*) may converge as a Lebesgue integral,and $\omega(z)$ may be replaced by $e^{iz.\zeta}$,giving a standard 'Fourier representation'.

In the case of a system of partial differential equations,$P_j(D)f=0, 1\leq j\leq N$,the (essentially)sufficient set, δ_P,is to be replaced by the cartesian product of the sets δ_{P_j} determined from the polynomials corresponding to the operators $P_j(D)$.
The solutions 'ignored' in obtaining this ,over-simple result are the so-called exponential polynomial solutions.For ordinary differential equations,these are associated with multiple zeros of the operator polynomials.In the present case, a proper treatment of the extra solutions requires the introduction of a new concept:that of 'multiplicity varieties(basically,collections of algebraic,or algebroidal,varieties,each with its own 'differentiation operations').Although this is an extremely technical subject,and it is far from clear that even the most basic calculations can be made fully effective,the potential power of these methods is so great,that 'experimental work' in this area must be worthwhile.The fact the support of the measure μ may be chosen to be an algebraic variety(at least,as a'first approximation) if T satisfies a suitable system of linear,partial differential equations,establishes a deep link between Ehrenpreis' methods and some aspects of constructive algebraic geometry(which is,already,of crucial importance in the development of algorithms for the integration of algebraic functions—see,Section 12,and Davenport(1979a)).Moreover,it turns out that,for the treatment of 'multiplicity varieties(and a limited extension of the theory to linear differential operators whose 'coefficients may be polynomials,local analytic functions,algebraic functions,or algebroidal functions)a very general study of algebroidal varieties is necessary.The occurrence of algebroidal functions in

algorithms for determining the 'small solutions' of (analytic)operator equations (see Section 15.8)is also suggestive of the potentially wide applicability of constructive procedures involving algebroidal functions and their associated varieties;and this adds weight tothe claim that such procedures merit intensive study,even though,superficially,they may appear unpromising as objects for use in symbolic analysis.The dominant problem is to develop effective formulations of all these concepts and techniques,so that they may be applied to concretely-represented systems of partial differential equations.This is a formidable task;but,the remarkable progress made by Davenport in the highly abstract field of algebraic geometry shows that the implementation of methods such as these is by no means impossible,and should be pursued seriously.All of the(expository and technical) material required to embark on this program may be found in Treves(1967),Ehrenpreis(1970),and Brenstein and Dostal(1972),which,together,give references to most of the original papers.

The operational due to Maslov are directed towards the solution of problems involving functions of <u>non</u>commuting operators,over Lagrangian manifolds.It is the systematic treatment of algebras of formal,multiple power series in noncommuting operators,and the development of a calculus for noncommuting operators, that that formthe most distinctive features of this work.Most of the current applications lie in fields of theoretical physics—and this is quite sufficient to justify efforts to introduce symbolic computation in this area,even if extensions covering other applications cannot be found.Apart from the questions of solvability(in the sense of <u>existence</u> of solutions)for various(systems of) (non)linear,partial differential equations,potentially effective methods for obtaining asymptotic representations of solutions are derivable from the general theory.In spite of its diversity of application(many subfields of physics are covered:e.g.,plasma physics,solid state physics,nuclear physics,and electronic optics),and of the possible levels of analytical sophistication at which it may be presented,the essential result for the entire theory may be formulated comparatively simply,in terms of so-called <u>quasi-inverses in algebras endowed with μ-structures</u>.

A μ-<u>structure</u> is analogous to(a particular specification of) Church's'lambda-symbol'(see Section 1).Each of these devices determines ordered n-ples of'variables',for substitution as arguments of functions(the main difference being that, the lambda symbol is intended to remove possible ambiguities in the mechanical assignment of variables to functions,as represented in a computer;whereas the μ-operation imposes a mapping of ordered sets of indeterminates into sets of noncommuting operators with specified 'locations';so that,in effect,an ordering of application of the operators,and a prescription of their mutual separation by collections of unit operators,is specified). In general,an algebra,A,is endowed with a μ-structure IFF

$$(\exists M \subset A) \,|\, \{\alpha_j \in M \wedge n_j \in Z^+ (1 \leqslant j \leqslant J;\ n_r \not\leq n_s, \text{unless } \alpha_r \text{ and } \alpha_s \text{ commute})\} \Rightarrow$$

$$\Rightarrow (\mu f)(\alpha)_J = [f(\langle\alpha_1;n_1\rangle,\ldots,\langle\alpha_J;n_J\rangle)] := [f(\langle\alpha;n\rangle)_J] \qquad (*),$$

where the operators α_i act in the reverse order to that of the n_k(which are, by convention,increasing in k),and $f(\langle\alpha;n\rangle)_J$ has the formal power-series

representation: $f(\langle\alpha;n\rangle)_J = \sum_{(i)_J} (c_i)_J \prod_{0\leqslant k\leqslant J-1} \{\alpha_k^{n_k} I_{n_{J-k},n_{J-k-1}}\}$ (**),

each symbol $I_{p,q}$ indicating a 'row of unit operators,filling the places between the p th and the q th'. Here,one works within the framework of an algebra of formal power series in(arbitrary)finite numbers of indeterminates(chosen from an infinite set),and an associated noncommutative algebra of operators.The square brackets create a 'barrier' against the imposition of ordering on operators inside them,as a result of any ordering imposed on terms outside them—that is,they are 'insulators against externally-imposed ordering'.Such brackets must be used in a full development of the noncommutative calculus.In particular,an axiomatic definition of the μ-operation may be given,and the axioms are adaptable for use in symbolic analysis.

To define 'quasi-inverse',let $A_{\langle\mu\rangle}$ be a (noncommutative) algebra with a given μ-structure,let $\alpha_1,\ldots,\alpha_J$,β,be operators from the algebra,and denote by L a module over $A_{\langle\mu\rangle}$. Then: f is a quasi-inverse element (with associated right- , and left-,quasi-inverse sequences, $\{S_k^+\}$, $\{S_k^-\}$,respectively,each contained in L) IFF $fS_k^+ = \underline{1} + \rho_k^+(\langle\alpha_i;i\rangle,\langle\beta;J+1\rangle)_J$; $S_k^- f = \underline{1} + \rho_k^-(\langle\alpha_i;J+2-i\rangle,\langle\beta;1\rangle)_J$, where,as functions,the $\rho_k^{\pm}((x)_J,\theta)$ are $O(\|(x)_J\|^{-k})$,as $\|(x)_J\| \to \infty$.Thus, given f,there exist sequences S^+,S^-,and 'remainders', ρ^+, ρ^-,interrelated as above. The principal problem in the theory is to find effective methods of calculating quasi-inverses in a variety of concrete settings.In these terms,the 'Main Theorem' may be stated as follows. Let α_1, , α_J, β, generate a nilpotent Lie algebra, and define the Hamiltonian function,H_Λ,of an operator $\Lambda = f_\Lambda(\langle\alpha_i;i\rangle,\langle\beta;J+1\rangle)_J$ to be the'leading term',say, $H_\Lambda((\gamma)_J,\eta)$,obtained when all of the α_i,and β, are given concrete representations(e.g.,in terms of 'position' and 'momentum' operators).Then (a) $\Lambda = f(\ldots)$ is a quasi-inverse element(whose associated right-, and left-,sequences are to be determined in terms of the α_i and β) IFF

IFF H_Λ satisfies certain'absorption conditions'(governing the 'time-evolution' of solutions of the 'Hamilton equations' generated by the 'Hamiltonian',H_Λ)— modelled on Sommerfeld's 'radiation condition for problems in electrodynamics. (These conditions may be specified precisely).

(b)Effective procedures may be given for the (approximate)construction of quasi-inverses.(Usually,this amounts to showin that,effective approximations to the quasi-inverse sequences of f_Λ may be constructed by solving 'sufficiently regular' integral equations—for which task a whole range of possible procedures is known, and accessible to symbolic computation).

In general,it may be shown that the determination of a quasi-inverse converts a given partial differential equation into an asymptotically equivalent(Fredholm or Volterra)integral equation,with a smooth kernel,decreasing 'rapidly at infinity'. This integral equation may be tackled by various methods,to produce approximate solutions of the original equation.The details of procedures for constructing quasi-inverses are very complicated,and involve(in general cases)extensive use of results for operator algebras and Sobolev spaces--and,even topological techniques. Nevertheless,the procedures are quite explicit,and potentially effective(since error estimates amy be obtained at each stage).A central rôle is played by analogues of Hamilton's and the Hamilton/Jacobi equations(in separating the solution into 'rapidly oscillating' and 'well behaved' parts),the 'Hamiltonian function' for a given operator being obtainable constructively from the representation of the original operator in terms of the generators of a suitable,nilpotent Lie group.(The operator 'β' which appears in all of the above definitions,corresponds to operations relative to various 'extra parameters',depending on the problem under consideration).
As for Ehrenpreis' scheme(indeed,even more so)there are enormous problems in disentangling the strands of Maslov's theory to fashion feasible approximation routines for use in solving certain types of partial differential equations involving noncommuting operators.In particular,the systematic treatment of multiple power-series of noncommuting operators requires(for symbolic analysis)a means of handling corresponding truncated series.Again,integrals involving operators must be applied to appropriate operands(e.g.,in the construction of quasi-inverses).Some work relevant to these matters has been done.The 'symbol calculus' studied by Voros(1977, 1979)includes valuable basic relations,and offers possibilities for adaptation and extension to the present,more general framework.Moreover,frequently,operators have(finite)matrix representations,and may be handled in this form without undue difficulty(even if their elements are,themselves,operators).The determination of concrete representations of given operators(in the sense requred to obtain their 'Hamiltonian functions')proceeds in several steps.First,the given operator,say,Λ, in a noncommutative algebra,is(formally)expressed as a function,say,f_Λ,of the

generators, α_i,of a selected,underlying,nilpotent Lie group(and,of the 'external operator, β).Next,the α_i themselves are expressed(using an effective procedure) in terms of prescribed(concretely realized)operators—after which,the resulting expression is put into a (well-defined)canonical form,where the leading term can be identified(again,in an effective way).This leading term is the Hamiltonian function of Λ.The absorption conditions refer to solutions of the 'Hamiltonian equations corresponding to H_Λ. Consequently,one basic symbolic computation routine would determine(as output)the Hamiltonian functions of operators given as input(along with all of the data required to convert operators to concrete form). Another essential routine would deal with calculations in Lie algebras(especially,with the formal manipulation of(iterated)commutators).Certain formal analogues of Taylor expansions(with 'remainder')must be covered,too.(Similar types of manipulations are necessary in many parts of mathematical physics—for instance, in the statistical mechanics of 'spin systems',and other 'lattice models';so, efficient symbolic computation routines of this kind could be of wide application, quite apart from their use in the present context).The basic analytical problem is to identify operators(formally expressed in terms of specified generaors) with symbols(symbols of 'rank k' being C^∞ functions on R^k,which,along with all of their derivatives,grow 'at worst,polynomially' as the norms of their arguments become infinite).Although Maslov's scheme is intended to be of fairly general use, it would not be easy to transform arbitrarily given partial differential equations into forms suitable for the application of the 'Main Theorem'.Nevertheless,the challenge of extracting from this maze of formulae and functional-analytic techniques,some comparatively transparent(and,effectively implementable) approximation procedures,is one that should be met,since such procedures would be of potentially wide application to problems of practical importance—and there are,apparently,no other approaches of comparable scope available.

15.1.6. Topological approximation of fixed points of mappings.

Brouwer's(1912) proof of his fixed-point theorem,for mappings between finite-dimensional spaces,was framed in a constructive manner(in line with his advocacy of effective procedures,and the criticisms of 'existence proofs',which grew,ultimately,into 'Intuitionism':see,e.g.,Beth(1968) for a readable account of these matters).However,variants of Banach's fixed-point theorem(based on the existence of 'contraction mappings'between normed spaces)with their concomitant error estimates,were,until recently,the only 'fixed-point results' used by numerical analysts—even though the premises for contraction theorems are more stringent than the premises for Brouwer's theorem,and for later,topologically-based results(see, e.g.,Edwards(1965),Smart(19)).The vast range of applications(in econometrics, game theory and nonlinear programming,as well as in the solution of (systems of) (non)linear algebraic/(functional-)differential/integral equations)makes it important to develop as many effective versions as possible of fixed-point theorems, for ultimate use in numerical calculations.On the other hand,the determinantion of symbolic expressions,giving rise to these numerical approximations,is of great interest for symbolic analysis--and may be accomplished in essentially the same way as for purely numerical computations.

The Brouwer theorem(which states that,if K is a compact,convex subset of R^n,and f maps K continuously into itself,then f has at least one fixed-point('fp'))was singled out by researchers as being of basic importance in any attempt to furnish effective proofs of the more general(weak) fp theorems.The first 'topological procedures' developed especially forthe approximate location of fp's of mappings of an n-simplex into itself,proceeded,roughly,by constructing a seuqence of simplexes,with diameters tending to 0,each containing the desired fp.Several variants of these methods were given(with the aim of increasing the efficiency of the basic algorithm).The crucial idea underlying current methods of topological approximation of fp,is that of homotopic transformation of the original problem,into another problem which is trivially solvable.More precisely,let f map R^n continuously into itself,where $f(R^n)$ is compact,and define $H_f: [0,1] \times R^n \rightarrow R^n$,by $H_f(t,x) := (1-t)\xi + tf(x)$.Then,$H_f(1,x) \equiv f(x)$,and $H_f(0,x) = x$ IFF $x = \xi$. In this context,if one defines $\mathrm{Fix}(H_f) := \{(t,x) \mid H_f(t,x) = x\}$,then it may be proven(Browder(1960))that $\mathrm{Fix}(H_f)$ contains a continuum linking $(0,\xi)$ to $(1,x^*)$-- $f(x^*) = x^*$.Apart from these 'simplicial fp algorithms,there are the so-called homotopy continuation algorithms(which have been used,amongst other things,to identify branches in multi-parameter bifurcation problems--see also Section 15.8).In these methods,the idea is to 'travel along a homotopy curve' in the set $H_f^{-1}(0) \subset R^{n+1}$--this curve being determined as the solution of an initial-value problem , namely: (*) $(\delta/\delta c)H_f(c(s))\dot{c}(s) = 0$; $\| c(s) \| = 1$ $(s \varepsilon R^1)$, $c(0) = \eta$,where the operation $\delta/\delta c$ denotes Frechet differentiation , and η is a point on the curve $\{c(s) \mid s \varepsilon R\}$--assumed known;or else,guessed. For purposes of symbolic analysis,equation (*)may be integrated using a quadrature formula 'in symbolic form',with rigorous error estimates.(In exceptional cases,some combination of the Risch algorithm(see Section 12)and other facilities,may produce an explicit, closed-form solution;but this happens 'almost never').If no branching occurs(i.e., if there is only a single branch of the homotopy curve,for all parameter-values) then,the fp may be located,approximately,by monitoring the values of $\| f(c(s))-c(s)\|$ until one of them is less than d (a preassigned 'tolerance')---the corresponding value of s,say s*,giving the approximate fp,c(s*).Both types of algorithms have been investigated intensively,by many people,and the versions currently in use have a sufficient degree of permenance to constitute a sound basis for approximate symbolic procedures.

An excellent survey of the theory and diverse applications of all of these methods, is given by Allgower and Georg(1980),on which most of the preceding remarks are based.They include many references to the literature,and emphasize the development of effective procedures(with some analysis of efficiency and stability).In particular,they present 'prototype algorithms' for each of the basic methods,with detailed consideration of various modes of triangulation,and,of techniques of pivoting(i.e.,changing(repeatedly)from one simplex to another,in the course of a calculation---which is equivalent to the replacement of certain columns of matrices corresponding to the simplexes).Moreover,several other algorithms are given ---in adequate detail for their adaptation to purely symbolic forms(indeed,they are,essentially,already symbolic:it is mainly a matter of implementing them,optimally,within a symbolic computation system). The routines required here would make use of several other procedures outlined in this paper(notably,for the manipulation of Frechet derivatives,the subdivision of simplexes,and the use of implicit function theorems).The ultimate aim is to incorporate effective forms of the theorems of:Brouwer;Kakutani;Leray/Schauder,and Borsuk/Ulam(and,certain,less well known results),so that any of them may be 'applied',as appropriate,in the course of an investigation(see Section 14).Some algorithms allow the simultaneous approximation of several solutions;while,others cover unbounded or non-compact mappings(subject to 'coercivity conditions',which ensure that the sequences of 'approximate fp' generated are bounded).The potential applications of effective routines for the determination of fp are diverse,and make this an attractive field for symbolic analysis.Most of the procedures may be studied in terms of (differentials of)mappings between finite-dimensional spaces;and there are close links with some aspects of singularity theory---for instance,the fundamental result of Sard(1942)(that,the set of 'critical values' of a differentiable mapping has relative Lebesgue measure 0),is invoked to show that for 'almost all'starting points,certain algorithms will converge,to admissible fp;and, (topological)degree theory is the basis for several nonconstructive proofs of fp results.Other useful references for this subsection(and for Section 15.13)are Balinski(1974) and Todd(1976).

15.17. Calculations in singularity theory.

The procedures envisaged here refer,primarily,to three aspects of singulariy theory.(a)The automatic performance of'determininacy and unfolding' calculations, for given 'input functions'(with applications to 'catastrophe models' of various phenomena).(b)The detailed analysis of 'critical points of functions on manifolds' (based,in part,on the use of 'Morse inequalities').(c)The calculation(and use)of 'integer-valued characteristics of vector fields'(e.g.,the 'rotation').As far as topic (a) is concerned,the essential step has been taken by Olsen et al.(1978), who have produced(numerical)computer programs to test(for 'strong',or'local', determinacy,and transversality)any input 'k-jet' involving polynomials,and non-polynomial Elementary functions,in several variables.Codimension calculations are done,too,and the corresponding 'unfoldings' are obtained.If determinacy cannot be established for a particular value of k,then another value is selected,and the procedures are repeated.The general situation is that some multivariable function,F,is given,and one seeks the minimal value of k for which the (Taylor) k-jet, $F|_k$,satisfies the determinacy conditions.When this value of k is finite(say,k*), so is the number of 'control parameters' required in an 'unfolding'of F(i.e.,in a characterization of all possible types of behaviour realizable by $F+\eta g$,for 'small η and arbitrary functions g').Poston and Stewart(1976,1978)give elaborate 'rules for determinacy and unfolding calculations'.(The work of Olsen et al.(1978) seems to be based on these rules,and a listing of their ALGOL programs is included,as an appendix,in Poston and Stewart(1978)).Although the programs given are numerical,there is much scope for symbolic computation in this area.(It is mentioned as a possible option in treating non-Elementary functions,but no systematic use has been made of it,so far).Again,the diverse applications discussed in Poston and Stewart(1978)are amenable to symbolic computation,provided that their associated 'models' are suitable for analysis using the basic 'determinacy routines'.Mathematically,what is involved here is a mixture of multivariable calculus and linear algebra(subject to 'rules of procedure and interpretation',based on deep results from differential topology—see,e.g.,Trotman and Zeeman(1975)).In practice,the extent to which the resulting modes of 'catastrophic behaviour' may be attributed <u>ligitimately</u> to the phenomena modelled,is a subject of some controversy(see,for instance,the review,in the Bulletin of the American Mathematical Society,of Zeeman's(1977)'selected papers' in this field).There are,moreover,many other parts of singularity theory(even for 'smooth maps')than those dominant in catastrophe calculations(see,e.g.,Golubitsky and Guillemin(1973) for an account of the main results);and it is in this broader framework of differential topology that potentially effective procedures for symbolic analysis should be sought(for instance in more general investigations of transversality,of various species of stability, and,of the classification of singularities,where,e.g.,much of the work of V.I. Arnol'd has,already a constructive flavour—offering interesting possibilities for symbolic analysis(see the bibliography in Poston and Stewart(1978)).

Topic(b) is concerned with the implementation and potential applications of those results known,collectively,as 'Morse theory',as a result of the fundamental work of Morse(1934) and the monograph by Milnor(1963),where many applications are discussed.Essentially,the topological structure of a manifold,M,may be related to the location,and type,of critical points of functions defined on M,through the so-called Morse inequalities,and other conditions.(For a remarkable application,to 'frequency density functions' in lattice dynamics,see van Hove(1952)).Much of the basic material here is required also for the catastrophe-type calculations;so,it would be possible to incorporate both of topics (a) and (b) in a single symbolic computation package.The(uniform)approximability of smooth,bounded,real-valued functions on M,by sequences of functions having only nondegenerate critical points, is a basic result with wide implications;so,an effective procedure for this mode of approximation would be one desirable aim.Another area of interest is the use of Morse theory to determine the 'homotopy type' of a manifold.Thesetechniques may be viewed as tests for homotopy-equivalence.As such,they are useful in applications—for instance,in relation to the routines outlined in Section 15.13,which are valid only for spaces homotopy-equivalent to polyhedra;but,they are are also far more generally applicable(and strong enough to establish,e.g.,that,any compact manifold admitting a differentiable function with only two critical points,is homeomorphic to a sphere).For the purposes of experimental symbolic analysis,a routine for analysing the 'level sets',$f^{-1}(w)$,for Morse functions,f,on a manifold,M,to obtain information about the topology of M,would form a useful basis for further work in this field.If this were successful,then,more advanced applications(e.g., some of those discussed by Milnor(1963),and by Morse and Cairns(1969))could be attempted.

The third type of singularity calculation considered here is centered on the concepts of index,of a singular point,and,rotation,for a vector field.The study of plane vector fields is especially interesting in the context of symbolic analysis, since there are many applications in function theory(e.g.,location of zeros;study of harmonic functions),and to (non)linear boundary-value problems—as well as,to the investigation of periodic solutions of n th-order (systems of)(non)linear equations.The use of topological ideas in function theory was pioneered(in its modern form)by Morse(1946),where references are given;but the applications(developed, mainly,by Soviet mathematicians)are of more recent origin.A readable account of work in this area is given by Krasnosel'skii et al.(1966),and there are many opportunities here for symbolic analysis.Indeed,most of the applications are framed in a 'quasi-constructive'manner,and it would not be hard to produce from them fully effective routines for symbolic computation.Moreover,many of these results have implcations for the study of phase curves('trajectories')of dynamical systems,and may be applied in conjunction with the methods mentioned in Section 15.12(on stability analysis of solutions of nonlinear differential equations).Among the basic routines for symbolic analysis would be those for calculating the rotation of a 'given' vector field;and,for finding theindex of a singular point in the field. Other rutines would cover various applications(as indicated above).Apart from the applications to relatively 'practical' problems in function theory,and in the study of differential equations,there are,also,investigations of a more fundamental nature,on the use of topological methods in (classical) analysis—see,e.g., Whyburn(1964),where references to the original papers are given.

Taken together,the three topics mentioned in this subsection offer wide scope for symbolic analysis—not least,because they have many applications to problems of practical significance.Lastly,one should remark that Krasnosel'skii's(1964) approach to bifurcation theory,for nonlinear integral equations(also based on the concept of 'rotation'),is an alternative to the more direct,analytical methods discussed in Section 15.8—and may complement them,in some circumstances;so,effective routines based on these topological criteria would be valuable,too.

15.18. Finite element (and related) methods.

This is a vast field(the computer-based bibliography maintained by Norrie and de Vries(1976-)contains over 7,000 citations!);but there are certain types of calculations for which fairly general symbolic routines could be developed.Indeed, some preliminary work in this direction has been done already,and this work will be mentioned later.Moreover,not only are there many levels of abstraction at which variational problems may be treated,but there are,also,two different approaches to the approximate solution of boundary-value problems--corresponding,respectively, to dissections of the whole domain(standard finite element methods),and,to dissections of the boundary of the domain('potential',or'boundary element' methods). Roughly speaking,the finite element schemes tend to produce large,sparse systems of (non)linear (algebraic) equations;whereas,the boundary element approach yields relatively small,dense systems of equations.Again,the matrix elements associated with these schemes are,in general,determined by integration--over the whole domain, or over its boundary;and this,too,is an important consideration,where numerical work is involved.

The rigorous approach to variational analysis makes use of general Sobolev spaces, 'weak solutions' being defined with the aid of suitable sesquilinear forms(see,e.g., Showalter(1977)).In the case of finite element methods for elliptic partial differential equations,a definitive account has been given by Ciarlet(1977);another good treatment,covering all types of boundary/initial value problems is Fairweather(1978). The general situation may be summarised,very briefly,as follows.(1)For certain (elliptic) boundary value problems,there exist genuine variational principles,giving the unique solution(e.g.,via the Ritz or Galerkin methods).In such cases,the weak solution(defined in relation to a bilinear form)coincides with the 'classical' solution.(2)Each solution is computed over some finite-dimiensional subspace of the space of exact(weak)solutions.(3)When the (boundary) data and the coefficients are 'sufficiently smooth',then the weak solutions are also classical solutions-- that is,they satisfy the partial differential equation,and the boundary conditions, and they are suitably differnetiable,in the classical sense(whereas,the weak solutions are,in general,(continuously) differentiable only in the distributional sense) (4)For each class of boundary conditions(i.e.,'Dirichlet','Neumann',etc.)there is a corresponding 'quasi-variational principle'(determined by an associated bilinear form,B,satisfying so-called coercivity conditions,which ensure convergence of the algorithms for approximate solution--and should not be confused with the coercivity conditions imposed in simplicial/homotopy fixed-point algorithms(see,Section 15.15).If these conditions are imposed over an'base space',S,then the unique solution of the boundary-value problem satisfies the condition $B(u_0,v) = F(v)$,for a suitable functional F,and for all v in S.(5)As indicated above,this formulation can be extended to cover even those cases where no genuine variational principle exists--and the resulting quasi-variational problem is solved by means of Galerkin's method(to specified approximations).The finite element method may be viewed, then,as an interpolation method,with associated approximation procedures.

Following Strang and Fix(1973),one may list the essential approximations involved in the finite element method,as follows.Interpolation of the original numerical data(typically,obtained from experiments);simplification of the geometry of the domain,by means of dissections into subdomains of simple shape(polygonal elements)—some of which may have 'curved parts',near the boundary of theoriginal domain,a possible cause of erro r.Next,choice of finitely many polynomial(or more general) trial functions,satisfying prescribed boundary(and,other)conditions,at the vertices (or,nodes)of the dissection;modification of the original boundary conditions,to fit the simplified domain;approximate evaluation of the integrals occurring in the underlying (quasi-)variational principle,on which the analysis is based—and subsequent approximation of the system of (non)linear equations to be solved..Finally, inluence of round-off (and other) errors on the actual solution of the resulting (non)linear system.Plainly,some(at least) of these sources of error could be removed if symbolic analysis were used both in formulating and in solving such problems.

Against this background,there are several types of investigation where symbolic analysis could be most valuable.(a)Routines for deriving the (quasi-)variational problems corresponding to 'given' (quasi-)variational principles.(b)Routines for producing (compatible) finite element dissections of specified domains—along with the associated matrix descriptions interrelating the local and global co-ordinate systems.The most common modes of dissection approximate the domain by unions of (hyper-)rectangles and (hyper-)triangles(with disjoint interiors).However,the resulting lack of accuracy near the boundaries can be significant;so,various types of 'curved elements' have been used(e.g.,so-called isoparametric elements—see,for instance,Strang and Fix(1973)).Estimates of the errors introduced by inadequate matching at the curved parts of boundaries,are given by Ciarlet,in Aziz(1972).

(c)More generally,as the demands for accuracy in solutions increase,it may be necessary to produce dissections fitting the boundaries far more precisely than has been demanded up to now.If the boundary is not of a simple geometrical form,it is by no means clear how such dissections can be effected(subject,also,to a variety of inter-element/boundary conditions on the 'solution' and its (weak) derivatives). One systematic approach to this problem,due to Wachpress(1975),envisages the element boundaries as 'algebraic curves'(to be determined),each element being associated,now,with rational—rather than,polynomial—trial functions.Although this scheme may be over-refined for many practical applications,it is of theoretical interest,because of its formulation in terms of basic concepts from algebraic geometry;e.g.,intersection properties of algebraic curves,resolution of singularities, and the determination of curves from finite sets of points on them.(This 'determination problem'—a generalized interpolation—has wide application.For instance, one method of conformal mapping of n-connected domains onto canonical domains,involves the approximation of a suitable Green's function,which is 'known' at a finite number of points(see,e.g.,Kantorovich and Krylov(1958),and,also,Section 15.7).This is almost identical to the problem encountered for 'rational finite elements').Among the results basic to Wachpress' approach are Bézout's theorem, and Max Noether's theorem(characterizing those curves passing through all points of intersection of two other curves).Algorithms are given for the construction of sets of basis functions,satisfying specified conditions on certain algebraic curves(parts of which coincide with inter-element boundaries).All of these procedures could be implemented for symbolic analysis--which would enlarge the scope of the method considerably,both in allowing boundaries of complicated structure, and in dealing with solutions that become singular at certain points.(Note that, when several different media are present,there may be internal boundaries,as well as the boundary enclosing the whole'system').

(d)Applications of symbolic computing to finite element analaysis have been made in a series of reports by Andersen and Noor(see,e.g.,Andersen and Noor(1977), where more references are given).They consider(for instance) the vibrations of laminated shells,and,by exploiting the underlying symmetry group,and using MACSYMA to evaluate exactly all integrals required in forming the matrix elements,they reduce considerably the total number of evaluations (since,repeated,numerical,operations are avoided).Further,they increase the accuracy of the solution(by postponing,as far as possible,all numerical procedures).The numerical calculations are facilitated by means of a (notational) device for translating symbolic code into (efficient) FORTRAN code.Naturally,if 'rational elements' were used,and a variety of problems investigated,it would not be possible,in general,to evaluate the integrals exactly(even with Risch's algorithm--see Section 12).Nevertheless, substantial gains in accuracy(and insight)should accrue to the use of symbolic computation,since,analytical approximations to integrals,etc.,could be used,anyway(giving much more information about the analytical properties of approximate solutions than could be obtained from purely numerical approximations).

One other significant step in the symbolic analysis of finite element procedures has been taken by Hall(1980),who has developed a collection of routines(using REDUCE)for obtaining symbolic representations of the 'Galerkin systems' corresponding to various types of boundaryvalue problems(over comparatively simple domains).Once again,all integrals are evaluated 'as analytically as possible',and there is,apparently,some hope of allowing more complicated domains in later versions.

Lastly,in this subsection,it is worth noting that there are many variants and extensions of the basic finite element approach--as originally developed for elliptic equations(where there is always a genuine variational principle,and the operators obtained are positive-definite).Among these variants are several 'collocation methods(see,e.g.,Fairweather(1978),and references given there).There are also several methods for the construction of trial functions--for instance, schemes involving the use of so-called spline subspaces.Here,too,there are many potentially effective procedures,suitable for implementation.For information on spline approximation,see,e.g.,Ahlberg(1967),and Rice(1969).Applications specifically to finite element approximation are given by Schultz ,in Schoenberg(1968).The most impotant facet of all these methods,as far as symbolic analysis is concerned, is that they permit rigorous error estimates,at all stages,so that,the symbolic approximations obtained in any calculation,could be appraised in the light of both their efficiency and their accuracy.

The boundary element method,referred to briefly,above,has become very popular recently,for certain types of problems,where the restriction of the operations to boundaries offers many computational advantages.Although there is,as yet,no fully rigorous analysis of this method(as regards convergence,sources of error, influence of irregularities in the boundary,etc.),the method is by no means new, since,it is an extension(suitable for numerical computing)of various 'potential-theoretic'methods,originating in electrostatics,and in elastostatics,where functions of interest are expressed as contour integrals involving 'density functions'-- which are to be determined.Usually,the end result is that the densities satisfy certain integral equations;and they are approximated using standard techniques for such equations.This approach is covered,at a fairly elementary level,by Jaswon and Symm(1977);and,at a much higher level--but with some obscurities--by Kupradze (196),who treats a number of three-dimensional elasticity problems;see,also,Kupradze et al.(1979),where a variety of three-dimensional problems in several areas of (static and dynamic)elasticity are covered in great detail,and'algorithms'for the solution of problems in thermoelasticity are discussed;the work of Oden(1972) is of interest in this respect,too.The boundary element method is covered,at an engineering level,in Brebbia and Walker(1980).Recall that Sherman's method(for the solution of the boundary-value problems of elastostatics over n-connected domains)is a boundary element scheme;and,that the Cauchy integral representations used by Muskhelishvili have much in common with this approach.It is only in the systematic dissection of the boundary,and in the use of 'weighted residuals'and similar, devices , that the boundary element method proper exists as a distinct approximation technique.Until it has been formulated(and analysed)rigorously,there will not be many opportunities for symbolic analysis--though,eventually,it could play a valuable rôle here.

15.19. Calculations involving bases in Banach spaces.

Many procedures which,in relation to finite-dimensional spaces,have a natural formulation,become problematical(or ill-defined) when the underlying spaces are infinite-dimensional.However,if the spaces are separable,and possess bases,then,more or less direct extensions may be given,of routines for the finite-dimensional case—and,such routines are amenable to symbolic analysis.Thus,this subsection complements Section 15.6,where 'calculus operations in general spaces' are discussed. Mostly,the concept of base is associated with Banach spaces;but there are possible extensions,to which reference is made later.There are several uses of bases in 'best-approximation theory',and these alone would justify the development of symbolic analysis routines in this area--but there are,also,many other applications.The encyclopedic treatise of Singer(1970a) offers a comprehensive treatment of theoretical questions about bases.The 'Volume II' referred to(in the Preface), has not appeared,up to now.It is intended to cover both generalizations of the notion of base—e.g.,to nonseparable Banach spaces,or to topological linear spaces—and,properties of bases in concretely represented Banach spaces(of special importance for symbolic analysis).For the present remarks,it will suffice to identify salient procedures,and to suggest a few possible applications.

A basis,in an infinite-dimensional Banach space,B,is a sequence,say, $\{x_n\} \subset B$, such that every x in B is representable,uniquely,in the form $x = \Sigma\, \alpha_i x_i$ (*), in the sense that $\| x - \sum_1^m \alpha_k x_k \| \to 0$,as $m \to \infty$.Here, $\{\alpha_k\} \subset K$(the'ground field'), and $\{\alpha_k\}$ is determined uniquely by x.The norm is prescribed in the definition of B.One question now arises naturally:given $\{y_n\} \subset B$,is there a decision procedure for the problem:'Is $\{y_n\}$ a basis for B ? '.?In other words,are there effective methods for deciding whether a given sequence is a basis in B?(Call this,problem (a)).A brief account of the most elementary aspects of problem (a) is given in Higgins(1978),which,although only touchng the surface of the subject,could provide some 'test routines' for initial computations.

(b)It is very difficult to establish basis properties 'from scratch';but,there are various results pertaining to the 'transmission' of(say) completeness,from a complete,biorthogonal sequence,to some (nonorthogonal) sequence 'sufficiently close' to it--in specified senses.Here,$\{x_n\}$ is complete if the set of all finite linear combinations(over K) of its elements,is dense in B (in the norm topology).Results of the 'transmission' kind are certainly implementable in symbolic analysis,offering a powerful tool for testing for completeness any 'input sequence'.Notice,however,that a complete sequence need not be a basis--even though,a basis always corresponds to a complete sequence.Nevertheless,there are characterizations of bases (and so-called stability results)which would make it feasible to construct routines for the identification of(some types of)bases--and this one important aim.Indeed, there are many results giving sufficient additional conditions for a complete sequence to be a basis--as well as,more general results,not assuming completeness. All of these procedures could be handled in symbolic computation.

Another 'obvious' routine,would accept as input any element,say, ξ,of B,and attempt to express ξ 'to order N',in terms of all explicitly-known bases for B. (Exceptionally,a 'general formula' might be found,for the k th coefficient in such a representation;but,mostly,the coefficients must be determined in turn--so that, only approximate representations,using partial sums,can be found,though,frequently, this is adequate for applications).In connection with completeness and biorthogonality($\{x_n\}$ is biorthogonal IFF $\exists\ \{f_m\} \subset B^* \mid \langle f_i,x_j\rangle = \delta_{ij}$,where the angular brackets denote the evaluation in the dual,B^*,of B,at elements of B) numerous criteria may be found to test whether a sequence is a basis,and many of these tests are potentially effective.For mutual orthogonality,certain variants of the standard definition are useful insome circumstances(and each of them has a simple geometrical interpretation).For example: $x \perp_1 y$ IFF $(\forall \alpha \in C)\ \|x+\alpha y\| \geqslant \|x\|$; $x \perp_2 y$ IFF $(\forall \alpha \in C)\ \|x+\alpha y\| = \|x-\alpha y\|$; $x \perp_3 y$ IFF $\|x+y\| = \|x-y\|$; $x \perp_4 y$ IFF $\|x-y\|^2 = \|x\|^2 + \|y\|^2$; and,lastly, $x \perp_5 y$ IFF $x \perp_3 y$ for all elements x,y,of unit norm.All of these definitions may be implemented straightforwardly for symbolic computation.Of course,approximations would appear in the evaluation of the coefficients α_k,as well as,in the restriction to partial sums--and the approximate evaluations would require the use of other symbolic analysis routines;but this raises no special problems.Several(interrelated) criteria for species of stability (for bases,and other sequences)may be implemented effectively.Examples include:conservation of linear independence in finite subsequences;or,of minimality;or,of completeness--all,in sequences 'close enough' to a given sequences,which is assumed to have these properties.Analogous results hold,also,for many other properties.Again,there are many special types of bases--each having certain defining characteristics;and all of these characterizations may be incorporated in a symbolic computation system--along with a collection of results for bases in concretely-represented Banach spaces(e.g.,the classical function spaces),which arise frequently,in practice.

The major use of bases to be discussed here,is their use in determining the 'best approximation(s) of order N' to a given element, ξ,of a Banach space,B,by linear combinations of elements chosen from a set $G \subset B$,where,either(i)G has finite dimension,say,n(so that,$G = [x_1,\dots,x_n]$);or else,(ii)G has finite codimension, say,m (so that,$G \dot{+} [z_1,\dots,z_m] = B$).There are various conditions ensuring the existence(and uniqueness)of elements of best approximation--see,e.g.,Singer(1970b), where an extensive theory is developed,combining techniques from measure theory and linear functional analysis in a way that offers many opportunities for symbolic analysis.

An element,y,in G is a'best approximation in G' to x IFF $\| x-y \| = \inf \{ \| x-w \| \mid w \in G \}$. The aim of the following procedure(in which a Banach space,B,admitting unique best approximation of elements,is given)is to construct a norm,say, $\| \ \|^{\Gamma}$,<u>equivalent</u> to the standard norm of B(in the sense that there exist constants,c_1,c_2, such that $c_1 \| x \| \leqslant \| x \|^{\Gamma} \leqslant c_2 \| x \|$,for all x in B)but having the property that the mapping,say, π_G,assigning to each element of B its unique best approximation in G,is <u>linear</u> (whereas,with respect to the standard norm,the mapping is,in general, <u>nonlinear</u>).This implies that,for purposes of approximation(to a given order) π_G may be 'replaced by a continuous,linear mapping of B onto G--in terms of which the calculation of best approximations(and,of elements of best approximation) is a simple matter.The key results are as follows(see Singer(1970a,pp175-6).

<u>The norm in</u> B <u>is a</u> 'T-<u>norm</u>'($^{T}\| \ \|$) with respect to a basis, $\{ x_n \}$, IFF (a)Every x in B has(for n = 1,2,...) a unique best approximation,say, $\pi^{[n]}(x)$, in $[x_1,\ldots,x_n]$;and,(b)this best approximation coincides with the n th partial sum,in the expansion of the element x relative to the basis $\{ x_k \}$.Analogously, <u>the norm in</u> B <u>is a</u> 'K-<u>norm</u>'($^{K}\| \ \|$) IFF (c)every x in B has(for n = 1,2,...)a unique element of best approximation in the 'complementary space', $[x_{n+1},x_{n+2},\ldots]$; and,(d)this complementary element coincides,for any n,with the 'remainder after n terms' in the expansion of x relative to $\{ x_n \}$.(Since equality of elements is defined to obtain when the norm of their difference is 0,or,tends to 0,for sequences,the necessity of changing from the standard norm to an equivalent T-,or K-, norm,is clear).The crucial point in this procedure is that there are potentially effective criteria for the construction of(or identification of) K- and T- norms; and it is such criteria which must be incorporated for symbolic computation.For instance,explicit realizations of these norms are given by:

(*) $^{T}\| x \| := \mathrm{Max} \{ n^{-1} \sum_{1}^{n} \| f_i(x)x_i \| + \| \sum_{n>1} f_i(x)x_i \| \mid 1 \leqslant n < \infty \}$; and

(**) $^{K}\| x \| := \sum_{i \geqslant 1} 2^{-i} \| f_i(x)x_i \| + \sup \{ \| \sum_{1}^{n} f_i(x)x_i \| \mid 1 \leqslant n < \infty \}$. Here, $\{ x_n \}$ is a basis in B,and $\{ f_k \}$ is the 'associated sequence of coefficient functionals', defined by the condition: $(\forall x \in B)\ f_k(x) = \alpha_k$ IFF $x = \sum \alpha_k x_k$.Since $\{ x_n \}$ is a basis,both of the terms inside the 'Max'sign tend to 0 as n tends to ∞;so, the maximum <u>is</u> attained,in (*).The situation for (**) is not so clear,and must be investigated,to see how far this definition is (approximately)implementable. However,there are many other criteria for T- and K- norms,which may be more suitable for suitable than (**) for use in symbolic analysis:there is wide scope for experiment,here.The main result may be stated,now,in a simple way.

Theorem.Let $\{x_n\}$ be a basis for B,and define $e_{x,N} := \inf\{\|x-p\| \mid p \in [x_1,\ldots x_N]\}$. Then,if c_1 and c_2 are the 'equivalence constants' for the standard and T- norms on B,and σ_k is the 'partial sum operator',taking each x in B to the k th partial sum in the expansion of x relative to $\{x_n\}$,in the standard norm,the following inequalities hold: $(c_1/c_2)\|x-\sigma_N(x)\| \leq e_{x,N} \leq \|x-\sigma_N(x)\|$.

Plainly,there are many 'contiguous' areas in which symbolic analysis routines could be developed;but,the object of this subsection is merely to show the feasibility of attacking problems of this kind,not to compile a large collection of examples.However,one other aspect of best-approximation theory is worth mentioning here,namely,the concept of near-best approximation,introduced,in the context of practical approximation procedures,by Mason(1970).For a review of some recent results,see Mason(1980).If an element,x,in a normed linear space, has a best approximation,say,x_n^B ,in a subspace of dimension n+1,then an arbitrary approximation,say,x_n^*,is 'near-best within a relative distance ρ_n' IFF $\|x-x_n^*\| \leq (1+\rho_n)\|x-x_n^B\|$ (***).If $\{\rho_n\}$ is a suitably decreasing sequence,then (***) has useful practical implications.Results of this type have been obtained for real L_∞ approximation;real L_1 approximation,some forms of real multivariate approximation,and for certain 'asymptotically near-best approximations.Moreover,a fundamental role is played by 'minimal projections'(if P_n projects B onto a subspace of dimension n+1,and if $\eta_n := \|P_n\|$,then it may be shown that (***) holds for any $\rho_n \geq \eta_n$),and several of the results are obtained from projections based on series expansion and interpolation procedures).Although this is intended to be,primarily,a numerical technique,there are many advantages in having at hand symbolic representations of near-best approximations(if more precise approximations are lacking).

Extensions of the notion of basis to linear topological spaces(in conjunction withthe use of proximites,uniformities,and other,general topological structures) would allowthe formulation of approximation problems,in a quasi-metrical framework, (and the generalization of may concepts),over a wide range of fields where, currently,very few 'quantitative results' are available.This is a project on which I have been working for some time.(One general approach to approximation problems in analysis—including the treatment of various types of boundary-value problems,systems of differential equations,and integral equations--is due to Stummel(1973,1975,1979),who is able to cover several aspects of numerical analysis in a unified way(mainly,in a Banach-space context).It may be possible to implement some of Stummel's techniques effectively,for symbolic analysis--for instance, criteria for convergence,stability and consistency.See Section 15.19).

15.20. Some calculations in rigorous statistical mechanics.

In statistical mechanics, the aim is to derive observable (equilibrium and nonequilibrium) properties of 'macroscopic systems', from their microscopic descriptions, in terms of interactions among the particles of which they are composed. Usually, this means that the Hamiltonian function is 'given' on the 'phase space' of the system, in the form: $H(\alpha) = T(p) + V(q)$, where T represents the kinetic energy, and V the potential energy, for a 'system' comprizing n particles, moving in a 'container', Ω, of volume $|\Omega|$ —T being a function of the n generalized momenta, p_j, and V, of the position co-ordinates, q_k. (It is possible, in some circumstances, for T and V to depend on both the p_j and the q_k; but this case is not considered here).

Although there is wide scope for symbolic computation in many parts of theoretical physics and chemistry (several of the computations packages described in this paper were developed, principally, for relativity (see Section 11), high energy physics (see, e.g., Campbell (1974)) and celestial mechanics (see, e.g., Barton and Fitch (1972a)), it appears that, rigorous statistical mechanics raises, in a natural way, a remarkable variety of deep mathematical problems. In order to identify these problems (even in the barest outline) a few definitions must be given. (Only 'classical' statistical mechanics is considered here. In 'quantum' statistical mechanics, various operator-theoretic questions arise, some of which may be amenable to symbolic analysis; but, these questions are too technical to be described briefly. However, certain matters can be considered 'in parallel', for the classical and quantum cases; see, e.g., Bongaarts and Siskens (1973)).

The partition function, for the system, say, S, specified above, is defined by:

$$(*) \qquad Z(\beta, N, \Omega) := (N!)^{-1} \int_{\Delta} e^{-\beta H(\alpha)} d\alpha,$$

where $d\alpha$ denotes integration over the phase space, Δ, of the system. This 'corresponds' to a system in thermal equilibrium with its surroundings, at temperature $(k\beta)^{-1}$, k being Boltzmann's constant. The grand partition function (for a system at temperature $(k\beta)^{-1}$ and fugacity, z—or, activity, λ—able to exchange particles with its 'surroundings') is given by:

$$(**) \qquad \Xi(\beta, z, \Omega) := 1 + \Sigma \lambda^N Z(\beta, N, \Omega) =: 1 + \Sigma (N!)^{-1} z^N Q(\beta, N, \Omega),$$

where, $Q(\beta, N, \Omega) := \int_{\Omega^N} e^{-\beta V(q)_N} d(q)_N$, is called the configurational integral.

Typically, $V(q)_N$ is a sum over pairwise interactions, which are cenrally symmetric:

$$(***) \qquad V(q)_N = \sum_{1 \leq i < j \leq N} \psi(|q_i - q_j|),$$

in which ψ is called the pair interaction potential. (Extensions covering many-body interactions, and potentials that are not centrally symmetric, are of importance in some applications; but they are not discussed here). The kinetic energy is expressible as a diagonal quadratic form in the generalized momenta;so,the integrations involving $T(p)_N$ may be performed trivially.This representation for $T(p)_N$ is appropriate in most cases of practical interest.Again,analogous definitions are obtained for the so-called lattice models(where the'particles' are either,spins,fixed at the points of a discrete space lattice,or else,or else,'itinerent particles',whose possible positions are assumed to be confined to such lattice-points).Typical examples are:models of (ferro-)magnetic crystals,and,of adsorbed gases.

The fundamental mathematical problems associated with this microscopic description of 'matter' may be summarised as follows.Denote by D* the general mathematical description outlined above.(P_1)Prove that the accepted thermodynamic properties (including the 'Laws of thermodynamics')are derivable from D*.(P_2)Calculate the 'equation of state' for S(a functional relation interconnecting pressure,volume, temperature,etc,for a system in overall equilibrium).(P_3)Prove that S 'tends towards an equilibrium state' if it remains 'isolated for a sufficiently long time'. (P_4)Account,mathematically,for the phenomena of changes of state('phase transitions'). (P_5)Characterize the possible equilibrium states of S.(P_6)Obtain adequate mathematical descriptions of the various states of matter(solid,liquid,dense gas,dilute gas,etc.).(P_7)Give a detailed mathematical description of systems'in the critical region'(i.e.,close to points of phase transition).I shall indicate now(ultra-briefly),how the problems,(P_i),encompass several interesting mathematical procedures-- many of which are potentially amenable to symbolic analysis.

(P_1) has two aspects:(a)to justify the replacement of 'time-averages' by 'mean values',computed using probability densities over phase space(the exact forms of these densities being very hard to derive rigorously).This is usually known as 'the ergodic problem':see,e.g.,Farquhar(1964),and,for a more sophisticated treatment, Mackey(1974).(b)Proof of the existence of the limit functions $\zeta(\beta,\rho)$, $\xi(\beta,z)$, given,respectively by the limits,as $|\Omega|$ tends to ∞,of $|\Omega|^{-1}\log Z(\beta,N,\Omega)$, and $|\Omega|^{-1}\log\Xi(\beta,z,\Omega)$ (where $N|\Omega|^{-1}$ tends to ρ,the 'particle density at infinite volume');and,determination of their analytical properties --e.g.,continuity, convexity,differentiability,analyticity--which,together,imply the laws of thermodynamics(see,e.g.,Fisher(1964),Ruelle(1969)).(P_1(a))does not offer obvious openings for symbolic analysis;but,in conjunction with (P_1(b)),it gives rise to interesting problems in asymptotic analysis.The point is that,since,infinitely large systems do not exist,it is highly desirable to study the asymptotic behaviour of the sequences of functions corresponding to the functions ζ and ξ;and,to combine this study with approximate evaluations of 'phase averages' for the associated, finite systems.Undoubtedly,this will generate messy expressions;but,one would expect to obtain series expansions(in N^{-1}, $|\Omega|^{-1}$,or some other 'small parameter')which could be handled,to fairly high order, with symbolic computation.One attempt to derive such expansions is due to Horowitz(1966,1968),whose papers would form a suitable starting point for these investigations.Another asymptotic method of some interest is developed in Iwata((1963),and later papers).

(P_2) raises questions of great complexity,and has been attacked on divers levels of sophistication.Perhaps the simplest 'self-contained' approach is that of Penrose (196),who introduces a 'Markovian postulate'—to the effect that successive 'states' of an isolated system(observed at times, $n\tau$,$n = 0,1,2,\ldots$) constitute a Markov chain.Thus,the tendency of the system to an equilibrium state is deduced from parallel properties for Markov chains.Of course,the basic postulate cannot be justified rigorously;it is suggested by physical considerations.Nevertheless,all other approaches,however mathematically intricate,embody <u>some</u> assumption:of an <u>analytical</u> nature(as in the representation of states of systems in terms of 'state functionals' on C*-algebras--see,e.g.,Robinson(1978));of a <u>probabilistic</u> nature(as in the imposition of various 'randomness conditions'—see,e.g.,van Kampen,in Cohen (1962),and,on a higher mathematical level,Davies(1977));or,of other types,not so easily classified.Although there are possibilities here for symbolic analysis,no potentially effective schemes suggest themselves directly (apart from the approximate solution of the so-called <u>master equations</u>,which,in some treatments,govern the 'approach to equilibrium' of the system,once the basic assumptions are accepted).

In (P_3),on the other hand,there are several lines of investigation where symbolic analysis could be most valuable.Roughly speaking,the equation of state should determine(by analytic continuation) whether a system can undergo phase transitions, and(if so),the nature of all phases in which it can (co)exist in equilibrium.This problem has been discussed,mainly,in terms of 'fugacity expansions'(power series representations of various functions),whose singularities are of crucial importance--see,e.g.,Katsura(1963);and,in terms of the distribution of limit points of (complex) zeros(in the z-Plane),of the grand partition function, Ξ ,as $|\Omega| \to \infty$.If, for instance,this distribution includes a point,say,x_0,on the positive,real-z axis, then there could be a phase transition at $z = x_0$ --with associated values of the other thermodynamic variables.(See,e.g.,Yang and Lee(1952)).In particular,for certain types of classical lattice models,it may be shown that all of the limit points of zeros of Ξ lie on a circle(Lee and Yang(1952))—a result that has been extended to many other lattice models(see,e.g.,Suzuki and Fisher(1970),Newman(1974),and Dunlop and Newman(1975)).It has been extended,also,to cover the limit points of zeros of Ξ in the complex β-Plane(see Jones(1966)).
In connection with these matters,certain problems arise.(i)<u>To form various fugacity or density expansions</u>(to prescribed order).(ii)<u>To perform the analytic continuation from a given equation of state</u>.(iii)To determine the limit-point distribution,for zeros of the grand partition function,corresponding to a prescribed interaction potential.

For (i),see,e.g.,the article by Stell,in Frisch and Lebowitz(1964),where techniques of functional differentiation and graph theory are used;and,Katsura(1963),for definitions and basic analytical properties.Detailed computations of such expansions (for both 'lattice' and 'continuum' models)have been undertaken by many people.One approach involves the use of Padé approximants--a technique that has been implemented for symbolic computation by Geddes(1978);and it is clear that the other methods,also,could be implemented(at least,partially).Graph-theoretic methods are used extensively,too(see,for instance,Ford and Uhlenbeck(196),and several papers by,e.g.,Domb,Sykes,Essam and Gaunt).Many of the graph-theoretic procedures are amenable to symbolic computation,and this may allow theexpansions to be carried further than has been possible so far.

Item (ii) is very difficult to investigate <u>generally</u>;but,one example of a detailed study,for the van der Waals equation of state,is given by Ikeda(1974a,b,c;1975). Although some aspects of his method need further justification,the whole study amounts to a description of the Reimann surface associated with the complete analytic function determined by the equation of state--together with various series expansions,representing the branches of this function in specified domains.This entire, compound procedure could be implemented efficiently for symbolic computation.For item (iii),the characterization,and determination,of he limit-point distribution is intimately related to the process of analytic continuation--though,no rigorously justified <u>general</u> prescription has been found,yet(this is a problem with Ikeda's work);but,see Penrose and Elvey(1968),and Elvey(1974),for a prescription valid for certain(essentially) one-dimensional models;and,Elvey(1973),for a proof that this prescription <u>fails</u> for the van der Waals equation--a fact that was conjectured by Penrose(unpublished)in 1967.An extension of the Yang/Lee results,due to Ruelle(1971),gives rise to a systematic computational scheme offering wide scope for symbolic computing(see,Runnels and Hubbard(1971),and Ruelle(1973)).

(P_4) is still very much open(insofar as the results prompted by P_1,P_2 and P_3 have failed to solve it).See,for instance,the contributions by Kac,and by Kasteleyn, in Cohen(1968),for accounts of this field;also,Baxter(1971a,b).(P_5) has been tackled,recently,in terms of the approximation of convex functions by tangent functionals(see,Israel(1975,1979)),and this procedure could allow explicit computation, in certain circumstances(possibly,in the construction of 'pathological' examples). There are,also,procedures for the decomposition of states into'collections of extremal invariant states'('Choquet theory'):see,e.g.,Lanford(197),and the lecture notes by Phelps(196).(P_6) includes the theory of lattice dynamics,various theories of correlation (e.g.,'long-range order') in liquids--involving all of the distribution function of orders $n = 1,2,\ldots$,for which systems of coupled integral equations must be solved(see,e.g.,Hill(1956),and Ruelle(1969),for a rigorous treatment) --a task in which symbolic analysis could play a very useful part,since,iterative techniques may be employed.Finally,(P_7) has been studied on the basis of the hypothesis that the'free energy density'function (ζ ,above)is analytic in both variables and so,is analytically continuable in both of these variables;and,similar assumptions are made for other'thermodynamic potentials'.By applying basic results from the theory of functions of several complex variables,Coopersmith(1968) obtains expansions for the potentials around critical values,and deduces 'scaling laws' (characterizing the singular behaviour of the thermodynamic functions within the critical region).This scheme is directly computational,and could be adapted for symbolic analysis.More recently,the so-called renormalization group approach (see e.g.,Wilson(19) and Barber(197)) offers rich opportunities for symbolic computing(though,it is difficult to make some of the calculations rigorous;and it may prove hard to cast them in algorithmic forms).

15.21. Approximate solution of (stochastic)(functional) operator equations.

The literature in this area is so vast and diverse, that, the only objective of this subsection is to identify a few types of calculations which are, already, essentially algorithmic; and so, could be implemented, fairly simply, for symbolic analysis. These calculations include: iterative procedures (for (non)linear, deterministic, (functional) operator equations), and methods for the constructive approximation of 'stochastic integrals', and, of the solutions to certain stochastic differential equations. The potential for symbolic analysis, here, is enormous, since, there are numerous, quasi-effective schemes, whose fully effective implementation has been prevented (with few exceptions) by the intractability of the computations involved-- even, 'to low order'. Naturally, various approximations must be introduced; but, it should be possible to derive suitable (analytical) error estimates in most cases of major interest. Throughout this subsection, many of the routines outlined in other parts of this paper are required--especially, those dealing with 'calculus in Banach spaces'.

For the approximate solution of operator equations there are many methods available. In particular, for linear equations, over (real, or complex) Hilbert spaces, several iterative schemes, capable of effective formulations, are discussed in detail by Patterson (1974). The variety of procedures is considerable, and includes some techniques potentially applicable to nonlinear equations, too. Thus, a sensible approach, for symbolic analysis, would start with a study of methods for linear operators; and then, coverextensions of some of these methods to nonlinear operators. After this, some of the techniques developed specifically for nonlinear equations could be examined, for effective formulations. Another basic source of techniques (for linear, and nonlinear operators on normed spaces) is Kantorovich and Akilov (1964)--a new (English) edition of which will appear very soon (published by Pergamon). In this work, there is a strong emphasis on effective methods. Again, Ostrowski (197) also contains much interesting material; and, a wide-ranging survey, covering various types of nonlinear euqations (e.g., integral equations, integrodifferential equations, delay-differential equations and difference equations) is given by Saaty (1967). Frequently, these methods involve the implicit function theorem (or, at least, similar hypotheses). Among the methods which could be implemented for the symbolic analysis of nonlinear equations are the following. (i) Altman's 'tangent hyperbola method' (characterized by:

(*) $x_{n+1} = x_n - Q_n \gamma_n T(x_n)$, where $n \geq 0$, $\gamma_n^{-1} := T'(x_n)$, $Q_n^{-1} := I-(1/2)\gamma_n T''(x_n)\gamma_n T(x_n)$,

and the primes denote Fréchet differentiation). (ii) Picard's method, for '$T(y) = y$, (with the iteration (**) $x_{n+1} = T(x_n)$)--useful for certain integral equations. (iii) Newton's method (where, $x_{n+1} = x_n - \eta_n f(x_n)$, for $\eta_n^{-1} := f'(z_n)$, z_n being chosen arbitrarily in some 'δ-neighbourhood' of the trial solution, x_0). (iv) The 'method of minimum error' (in which, $T(x) = 0$ is to be solved, and, $x_{n+1} = x_n - \Lambda_n A_n^* T(x_n)$ (***), where, $(A_n^* T)(x_n) :=$ grad $< T(x_n), T(x_n) >$, $\Lambda_n := \| T(x_n) \|^2 / \| (A_n^* T)(x_n) \|^2$, and the error estimate, $\| x_n - x^* \| = O(\lambda^{n/2})$ holds--x^* being the exact solution,and λ,a calculable,positive number less than 1).(v)The 'method of minimum residuals'(in which $x_{n+1} = x_n + \theta_n y_n$--for effectively calculable sequences, $\{\theta_n\} \subset R$,and, $\{y_n\} \subset H$,the underlying space).(vi)Ritz/Galerkin methods,(see Section 15.19),where some generalizations may be obtained if the bachground space admits a <u>basis</u>(see,also,Section 15.20).All of these techniques are valid only under certain additional conditions (mostly,referring to Frechet derivatives of order $j = 1,2,$or 3),and one essential task for symbolic analysis is to satisfy these conditions,<u>approximately</u>--but,in such a way that adequate(symbolic)error estimate can be derived.(Although there will be cases where this goal is unattainable(so that,any approximate implementation could be judged only by 'practical results'),it appears that the necessary estimates <u>could</u> be obtained,in many situations of importance for applications.For instance,when <u>separable</u> spaces are involved,linear operators have(generally,infinite)matrix representations,finite sections of which may be used.In the evaluation of Frechet derivatives,inverses of operators,etc.,analogous approximations inevitably appear;but,all of the methods mentioned here are,nevertheless,amenable to symbolic computation--in the sense that,expressions can be constructed,which satisfy the original operator equation(s),to given orders in suitable parameters;with the possibility,<u>in principle</u>,of indefinite improvement towards the exact solution(s), in cases where there is at least one exact solution.All of the specific classes of equations disussed by Saaty(1967)could be formulated effectively(in the sense just explained),and,an extensive collection of symbolic analysis routines could be assembled,by implementing these techniques,approximately.There are,of course,various other methods,especially,for nonlinear 'evolution equations',and integral equations; see,e.g.,Lattes and Lions(1969),where effectiveness is a high priority;and,Prodi (1971),for some examples of solution techniques.Here,again,the scope for symbolic computation is wide.

Assuming that effective methods can be found,for various types of deterministic operator equations,the problem arises of adapting some of these methods to handle equations in which certain,stochastic,elements are present.Once again,no attempt will be made to treat this matter comrehensively--even,in outline.Instead,two problems will be considered(with a view to effective symbolic implementation),namely: (a)<u>the approximate evaluation of stochastic integrals</u>;and,(b) <u>the approximate solution of certain classes of stochastic differential equations</u>.Moreover,for each of these topics,the identification of one or two 'pre-algorithms' is the only aim(as a prelude to more systematic investigations).

<u>Stochastic integration</u> (also known as 'functional integration') originated in Wiener's studies of Browian motion(Wiener(1923)).In much the same way as 'Haar integrals'and 'Haar measures' can exist separately,as well-defined objects,without their being a universal procedure for'associating them in pairs',it is possible to define various measures,in terms of stochastic processes;and,to find conditions sufficient

for the existence of integrals over these measures.For the Wiener measure,an explicit formula may be obtained for the integral (e.g.,for bounded,continuous functionals,F,on the space,X,of continuous functions,x,on $[0,\gamma]$,with $x(0)=0$),namely:

$$(*)\qquad \int_X F[x(\tau)]\, d_W x \sim (\pi\lambda/n)^{-n/2} \int_{R^n} F^*(x)_n \exp\{-(n/\lambda)\,\Sigma(x_{j+1}-x_j)^2\}\, d(x)_n,$$

as $n\to\infty$,where,$x_0\equiv 0$,and,$F(x)_n$ is obtained from $F[x]$ when the 'graph of x' is replaced by an n-gon,with its vertices on the graph,with ordinates $x_i, i=1,\ldots,n$.

This is,apparently,a most unpromising formula for effective approximation.Nevertheless,since the existence of the Wiener integral is assured(from general results),it is certainly possible to use (*) for 'large,finite values of n'(with suitable approximations for the integration over R^n,which,again,may be justified by the known convergence of this integral).In exceptional cases,(*) may be evaluated exactly(see, e.g.,Montroll(1952),Kac(1959)),and it would be quite feasible to incorporate all special results of this kind.In other cases,the 'value' of the Wiener integral is shown to be related to the solutions of certain parabolic evolution equations(the special results just mentioned corresponding to ordinary differential equations). Moreover,many results on the approximation of 'Feynman integrals' have been obtained in quantum mechanical investigations,and these,too,would be enhanced greatly,if they were formulated for effective computation.(See,also,Edwards and Lenard(1962)). More generally,in a series of basic contributions to this field,Cameron and Martin (1944,1945a,b,1949),and Cameron(1951,1954),laid the groundwork for a systematic approach to the approximate evaluation of Wiener integrals(and,to their manipulation;allowing,e.g.,changes in the 'variable of integration'),and a set of symbolic-analytic routines could be developed from this(and more recent)work.A good review, (with applications to quantum physics)is given by Gel'fand and Yaglom(1960),where several references to early papers in this field are given.For other types of stochastic integrals,there seems to be no source of 'pre-algorithms as direct as those for basic Wiener integration;however,the work of Young(see Ney(1970),Ney and Port (1974)) indicates that effective procedures could be developed to cover a wider class of stochastic integrals(and,some of the techniques used in studying stochastic differential equations are also relevant to the definition of new types of integral.

In 'stochastic differential equations',indeterminacy can enter in many ways(for instance,in:random initial/boundary conditions;in random coefficients;and,in random forcing functions).This comment applies,with suitable modifications,also to stochastic operator equations of all types;but,only differential equations are considered here.Again,one may study ordinary,or partial,stochastic differential equations;but the differences between these two cases are not due,substantially,to the presence of indeterminacy;so,it will suffice to consider stochastic ordinary differential equations(except in a few remarks,and references).For experiments in symbolic analysis,the 'mean-square convergence' approach of Jazwinski(1970),which avoids the use of measure theory,would provide a convenient starting point,the generic equation studied being: $(d/dt)x_t = f(x_t,w_t,t) =: f_t$ (**) ,for $t \geqslant t_0$,to determine the unknown n-vector,x_t,subjected to stochastic irregularities through the occurrence of a random 'disturbing function',w_t,as an argument of a specified,(non)linear function,f.If $\int_{t_0}^{t} f_\tau d\tau$ is well-defined(say,in a mean-square sense),then,(*) is equivalent to the relation: $x_t - x_{t_0} = \int_{t_0}^{t} f_\tau d\tau$ (**),and, $\{x_t\}$,given(implicitly)by (**),may be studied as a stochastic process,in its own right.One of the most practically significant cases of (*) is obtained if f splits into a deterministic part,say,g,and an additive 'white Gaussian' part,proportional to W_t,giving(formally): (***) $(d/dt)x_t = g(x_t,t) + Q(x_t,t)W_t$,where Q is an (n×m)-matrix,and,x_{t_0} does not depend on W_t.Since it may be shown that, $\{x_t\}$is mean-square(Riemann) integrable over an interval,J,IFF the 'covariance function of $\{x_t\}$ is integrable(in the deterministic sense)over J×J,it follows(from the properties of the white Gaussian process),that, $\int Q_t dW_t$ cannot be defined in the mean-square sense.To avoid this difficulty,one may 'replace' W_t (again,formally) $(d/dt)\beta_t$,where, $\{\beta_t : t \geqslant t_0\}$ is a vector process of (independent)'Brownian motions'.If this is done,then (***) is 'converted' into the relation (***)': $x_t - x_{t_0} = \int_{t_0}^{t} g_\tau d\tau + \int_{t_0}^{t} Q_\tau d\beta_\tau$,both integrals being well-defined,now.The second('Itô')integral in (***)' already requires an extension of the methods mentioned above,for Wiener integrals.The Itô integral is studied quite extensively(along with corresponding 'Itô differentials) in Jazwinski(1970)with several examples,and a number of potentially effective procedures(some of which are due to Wong and Zakai(1965a,b,c)).In addition,it is shown that the process corresponding to (***)' is Markovian,and that its transition probability density function satisfies the 'Kolmogorov equation'(also called the 'Fokker/Planck equation,in statistical mechanics).Jazwinski applies these,and related,methods,systematically,to problems of (a)(non)linear filtering,and(b)(non)-linear prediction(i.e.,sufficiently reliable determination of the state of a stochastic system,(a),instantaneously,and,(b),in the future--in each case,using the past and current observations.A retrospective modification of observations,in this context,is called smoothing). Although the aim of filtering is to obtain numerical results,it is likely that symbolic computation could be used to improve the quality of these results—by postponing the numerical operations to the final stages of a calculation.Moreover,there are numerous approximation problems associated with filtering theory,and here,also,symbolic analysis could be very useful.A somewhat more rigorous formulation of the Wong/Zakai approach to the effective solution of stochastic differential equations is given by McShane(1972),who outlines a 'Runge/Kutta' procedure(involving 'averaging' and recursion)for the approximate solution of such equations.Plainly,there is almost unlimited scope for the development of constructive symbolic analysis routines,in this area.

15.22. Miscellaneous topics.

(a)Some calculations in general perturbation theory.

This is precisely the kind of filed where symbolic analysis can prove most valuable, provided that specific(effectively realized)algorithms are developed.The necessary analytical estimates and convergence criteria are available(for perturbations involving general operators) in a Banach-space framework--which can be handled,for symbolic computation,in various contexts.For operators on finite-dimensional spaces, a complete theory exists(see,e.g.,Kato(1976),chapters I and II),and this may be implemented almost totally,depending,as it does,on a function theoretic analysis of resolvent operators--and,allowing estimates for rates of convergence of perturbation series.Moreover,the central problem(investigation of the behaviour of eigenvalues and eigenvectors,for families of linear operators depending analytically on a parameter)is related closely to aspects of bifurcation theory(see,for instance, Arnol'd(1971;and,Section 15.8).Matters covered by the'finite-dimensional theory' include:determination of singularities of eigenvalues(which may be represented by Puiseux series--see,also,Sections 15.1 and 15.8);determination of Neumann series for the resolvent,say,R;manipulations involving the projection operator,$P(\varkappa)$,defined by $P(\varkappa):=-(2\pi i)^{-1}\int_{\Gamma} R(\zeta,\varkappa)d\zeta$ (which 'equals' the sum of all the eigenvectors corresponding to eigenvalues of the basic operator,say,$T(\varkappa)$,lying inside the contour Γ),and manipulations for locating the singularities of the 'eigenprojections';calculations with perturbation series--e.g.,application of estimates for radii of convergence,and,for 'remainders after n terms';treatment of nonanalytic perturbations(e.g.,when $T(\varkappa)$ is merely continuous)--incorporating criteria for differentiability,in $\varkappa$,of eigenvalues and eigenvectors,at isolated points,or over domains;asymptotic expansions of eigenvalues and eigenvectors--and,treatment of these objects as functions of T itself,rathan,of $\varkappa$,T being represented by its matrix realizations.All of these facets of perturbation theory for finite-dimensional spaces are ideally suited to symbolic analysis;and,one very practical field of application is the numerical analysis of matrices.Several of the procedures just listed have fairly direct extensions to operators between infinite-dimensional spaces,and amny new problems arise in this,more general,case.For symbolic analysis, an excellent start could be made by implementing a comprehensive set of routines dealing with the finite-dimensional case;and then,examining possible extensions of these routines.Applications to problems in quantum mechanics could be contemplated(so that,some of the methods to be found in the extensive literature of this field could be implemented to comparatively high orders).Plainly,the longterm possibilities for symbolic computation in this area are diverse--though,the more abstract procedures would require very careful analysis,before they could be implemented effectively.

(b)Some calculations in abstract harmonic analysis.

The type of algorithms envisaged here, are concerned,primarily,with the construction of Haar measures(in various contexts) and on the determination of corresponding (Haar)integrals.Underlying this scheme,are certain constructive procedures,involving toplogical groups,group representations,group characters,etc.,which must(as far as possible) be considered for(approximate)effective treatment.The ultimate aim,for a restricted project,might be to obtain a 'noncommutative Risch algorithm',for the determination of integrals from given measures on topological groups(but,allowing for approximate constructions,as well as exact determinations).Thus,one routine would attempt to construct invariant measures on specified topological groups(with suitable conventions for the interpretation of approximate results).Then,a related routine could be used to form 'integrals',with respect to these measures(where this is possible),of functions defined on the specified group.The 'Haar integral' is a quasi-linear functional,defined on the group.(A fully linear integral may be obtained by using an extension procedure,making the domain a linear space).There are constructions for associating,with a given Haar integral,certain (left-,or,right-) invariant set-functions (or,'measures').Consequently,there are two('mutually inverse') problems:given a functional satisfying the defining conditions for a Haar integral,find a corresponding(invariant) set-function;and,conversely,given a measure function,construct a Haar integral from it.General(analytical)forms can be found, for the Haar integral over some classes of groups;and,manipulations based on such characterizations may be implemented for symbolic analysis--along with a collection of explicit results,where the Haar integrals can be evaluated as a (multiple) Riemann(or,Lebesgue) integral.(Some results of this kind are given in Hewitt and Ross(1963)).All of these explicit methods of computation could be incorporated in a symbolic computation procedure.As a second project,one might attempt to implement a few constructions involving convolutions and group representations.(In a different context,many techniques for obtaining representations for finite groups are capable of effective formulations,and this is another area where symbolic analysis could play an important part).On a more practical level,there may be calculations in quantum mechanics,or in high energy physics,where (e.g.) integration over Lie groups is required;and there are,certainly,potential applications in many parts of pure and applied mathematics.However,before embarking on elaborate schemes,one must experiment,thoroughly,with the basic calculations,until the optimal approach(for symbolic analysis)is found.The crucial point is,that this(predominantly,abstract) field should not be regarded as 'inherently unsuitable for symbolic computation'.On the contrary,routines of this kind would constitute a valuable complement to several of the other procedures outlined in this paper--and,ultimately,it may prove possible to develop a set of approximation schemes with much wider application than the ones tentatively discussed here.

(c)Calculations in computational geometry.

Computational geometry,as a well-defined field,is of very recent origin.Essentially, what is involved is the recasting of certain geometrical problems in computational forms.Among these problems are:the construction of convex hulls (of finite sets of points,in R^k);the determination of intersections of sets of points ('discrete',or 'continuous');and,the solution of 'closest-point',and,'searching',problems.Much of this work has wide practical application(e.g.,in cluster analysis,computer graphics, printed-circuit design,linear programming,pattern recognition,and applied statistics). There are,also,corresponding problems in higher-dimensional spaces(as indicated for the 'convex hull' problem),which are,at present,mainly of mathematical interest,but could find unexpected uses.(Some questions of these types even make sense in,say, Banach spaces--where there are potential applications,e.g.,in game theory,and in statistical mechanics).Only two possible sources of pre-algorithms are mentioned here:(i)the use of so-called geometric transforms (e.g.,in the construction of 'fast' algorithms);and,(ii)computational statistics.Source(i) has been developed by many people(especially,members of the Department of Computer Science,Carnegie-Mellon University).In particular,the doctoral thesis of Brown(1979) gives an interesting survey of the whole field(with many references),and it includes several specific transformations,together with an analytical approach to the determination of transforms---all of which purview of symbolic analysis.There is some emphasis on 'worst-case estimates'(i.e.,on finding upper bounds for the time,storage,or cost,required to produce results)---e.g.,for 'planar diameters';'intersections and unions'(in R^k); 'nearest,and farthest,points';and,'searching over tessilations'.A summary is given of all types of transformations so far found to be useful in this kind of work,namely:'point-to-point','duality',and,'others'.There are many problems in which it would be useful to have symbolic representations of the 'objects' and results obtained in computational geometry,and the underlying mathematical framework is,already,accessible to symbolic computation.

Source (ii),although it stems from geometric transform ideas,has become almost autonomous,since,itranges widely over statistical problems,and includes many results in 'applied complexity analysis'.The connection between geometry and statistics becomes clear when a 'sample' is regarded as a point set (in some Euclidean space).This field has been developed,principally,by Shamos(1977,1978).Various 'tendencies',of empirically determined distributions of points,may be characterized by associated parameters(e.g.,'skewness'),and the design of efficient algorithms to calculate such parameters(which may be viewed as real-valued functions of sets)is of importance.Moreover,the complexity of basic statistical operations may be found(or,located,within bounds);and,similar analyses may be given for the calculation of certain 'statistics' of practical importance.In the opposite direction,probabilistic arguments may be used to analyse 'average geometric behaviour'.(E.g.,Renyi(1963) shows that,if N points are chosen,'independently and uniformly' in a bounded,convex,plane figure,F,then,the expected value of the number of vertices in their convex hull,is equal to: $(2r/3)\log N + O(r)$,if F is a convex 'r-gon';and, $O(N^{1/3})$,if F has a continuously turning tangent.If,instead,N points are chosen from a planar,normal distribution,then,the corresponding expected value is $O(\sqrt{\log N})$.This is a fascinating area for symbolic analysis.

(d)Probability theory and mathematical statistics.

In this enormous field,there are almost unlimited opportunities for symbolic analysis. One has only to peruse such works as Feller(1957,1966),Renyi(1970),and Kendall and Stuart(1958,1961),to discover a plethora of pre-algorithms.In view of this situation, no attempt is made here even to list the most substantial routines which might be implementable.It is enough to remark that,the existing facilities for general linear algebra,and for classical analysis(in R^k),would allow the construction of efficient routines,covering the bulk of 'everyday calculations' in applications of probability theory and statistical analysis.In particular,most parts of the study of Markov chains are amenable to full symbolic-computational treatment;which is of great importance,since,apart from their many practical applications,Markov chains act as a basis for several,apparently unrelated theories(for instance,in the statistical mechanics of systems 'approaching equilibrium'(see Section 20);and,in the analysis of 'light fields',in Radiative transfer theory(see,e.g.,Preisendorfer(1965)).On a more abstract level,problems of asymptotic approximation(especially,in relation to variants of the 'Central Limit Theorem'--see,e.g.,Petrov(1973) ;in calculations involving random series of functions(see,e.g.,Kahane(1968) ;and,in the limit-analysis of 'convolution powers'(see,e.g.,Bergstrom(1963)) offer attractive possibilities for symbolic analysis.No serious questions of principle seem to be raised by any of these calculations:their difficulty resides in the extraction of efficient algorithms,and error estimates.

(e)Calculations in analytic complexity theory.

It is,perhaps,fitting,to end Section 15,which has ranged so widely over mathematics and its applications,with a few remarks on the possible use of symbolic analysis in estimating the analytic computational complexity of general classes of calculations--about which users of symbolic computation systems could be uncertain,because of the difficulty in assessing the storage demands involved.A major contribution to this field has been made by Traub and Wozniakowski,in a series of investigations(see,Traub and Wozniakowski(1980),where a large,annotated bibliography is given).In this work,they develop a general theory of optimal-error algorithms and analytic complexity,based on the concept of 'the information needed to solve(to prescribed accuracy)a given class of problems'.In practice,these 'problems' correspond,mostly,to the solution of various types of (nonlinear) operator equations;but, the approach is of great generality,encompassing many parts of classical approximation theory,as well as,the solution of equations over Banach spaces,of boundary-value problems,and,the analysis of quadrature procedures.In several cases,effectively calculable bounds are derived on the complexity of algorithms in which the data have specified 'information content'.The information may be of a general kind; but the most complete results are obtained in the case of 'iterative information' (on the basis of which,iterative schemes may be formulated for the approximate solution of the problem in question).The framework for most of these analyses is that of functional analysis in Banach spaces--and,the 'characteristic parameters', and associated bounds,could be approximate effectively,in many cases of practical significance.Although the basic motivation for complexity analysis is the estimation of cost(for various types of computations),this is related,intimately,to the determination of 'worst-case storage demands',which,if too high,would cause programs to stop.Moreover,such estimates(especially,in relation to optimal-error/optimal -efficiency algorithms) of defining 'efficiency'(for symbolic computation packages), and 'expressive power'(for symbolic computation languages);see,Section 16,for a few observations on these matters.Thus,in the area of complexity analysis,the primary aim is to design routines which can accept,as input,suitable specifications of calculations to be attempted,and,of the'information' to be supplied—the corresponding output comprising (symbolic) estimates of the complexities(which can be converted to quasi-numerical forms,whenever concretely-specified operators are involved.The implementation of such routines for symbolic analysis,though requiring careful preliminary research,is,still,perfectly feasible;and it raises fascinating possibilities(such as,calculating the complexity of algorithms for calculating the complexity of algorithms),which are,however,best left for a future investigation!

16. Remarks on 'efficiency' and comparison.

This is an area fraught with semantic,as well as mathematical,difficulties.'Efficiency' is a concept normally associated with 'machines';so,one would not attempt to define the efficiency of a symbolic computation language,but,rather,of a package, comprising a language,and its implementation(s).Here,already,a problem arises,since, different implementations of the same language may differ,also,in their 'observable characteristics'.One aim,then,is to define efficiency as an intrinsic property of a package(and,to define,say,'expressive power',for a language).For algorithms,the notion of efficiency has been studied extensively(see,e.g.,Karp(1974),Traub(1976, 1977),and,Traub and Wozniakowski(1980)).A possible approach to the definition of analogous concepts for languages and packages ,may be characterized through(respectively),minimal specification,and minimal realization(in terms of total storage used,total number of 'elementary operations'performed,etc.),of suitable classes of algorithms.For the purposes of symbolic analysis,these classes should contain all of the algorithms for basic symbolic procedures,together with as many of the more sophisticated algorithms as are fully specifiable in the language(or,realizabl in the package) concerned.It appears that this tentative prescription could be made precise if one proceed along the following lines.Every algorithm has a linguistic specification,involving basic symbols,predicates,and various constructions. Many definitions could be given of 'expressive power'(in this context):for instance. the minimal number of 'lines'(or,of 'simple statements') of program,required to specify all of the algorithms in an 'approved,fundamental set'— for a variety of types of data,covering the most common situations;or,alternatively,to specify only the most basic algorithms,from which the more sophisticated procedures are,ultimately,constructed.This is,of course,a 'gross' characteristic of a language,and it would have to be complemented by other,'finer',characterizations.However,it is clear that,in principle,many different measures of expressive power could be constructed,and that,eventually,a set of parameters could be associated with each language,in such a way as to summarise the 'bulk behaviour' of the language,relative to the writing of a wide range of programs.This information could be useful to designers,as well as to users,since,they could aim to produce an 'optimal set of parameter-values,in relation to a particular collection of 'calculation/data pairs'.In short,only the efficiency of algorithm/data pairs can be considered as a notion having practical value;and,such pairs may be analysed linguistically(to some extent) in isolation from their realizations.

For the corresponding definitions of efficiency for packages,it is necessary to decompose all of the algorithms in a fundamental set,into 'components',for which reliable(estimates of)operation-counts(or,other cost/time estimates) are available. There is no unique definition of 'elementary operation',but,one possibility,is to mirror the approach used in the design of algorithms for formal integration(see,e.g., Ritt(1948,Risch(1969),and,also,Section 12;Jeffreys(1961),and,Isaacson and Keller (1966),also consider the problem of assigning 'weights' to basic algebraic/analytical operations). The idea is to specify a hierarchy of function types,each,more (structurally) complicated than its predecessors.All types together(through composition and field operations) must encompas the full collection of functions of interest.Once this has been done,it should be possible to decide which operations to label as 'elementary',and to assign them weights;so that,if the numbers of elementary operations inherent in some calculation/data pair has been determined (or,estimated),this may be correlated with the minimal specification(or,realization) for the pair,in a given language(or,package).The definitions of 'power',and 'efficiency'must,somehow,combine information about the complexity of calculations,and, about the minimal prescriptions for implementing them.Some interesting ideas are contained in Knuth(1969),where the basic field operations are subjected to a searching analysis.

An important point already hinted at,is that,one must characterize the *data* for various classes of computations,in such a way that efficient algorithms *for given types of data* are used in estimating the efficiency of computation packages.The only absolute measure of efficiency for an algorithm(in isolation from the data on which it is to be used) must be based on its 'worst-case performance--which is not of much general use.Another basic requirement is that one should be able to *predict* the performance of a system,without doing all of the calculations contemplated.Thus,the 'efficiency parameters' associated with any specific computation *package*,can,at best,reflect some 'average mode of its behaviour',over a wide range of calculations,and types of data.To complement such 'mean-performance indicators', one needs other,'localized' criteria,applicable to individual computations(or else, to narrowly-defined calculation/data classes).Consequently,intrinsic characterizations of efficiency,etc.,though raising many interesting problems in mathematics and computer science,have to be studied more systematically,before they can produce results of much practical use.However,the combination of analytic complexity theory,mathematical logic,optimal-program design(see,e.g.,Schaefer(1973),where the underlying ideas stem from graph-theoretic analysis),and,the 'minimal representations',mentioned here,should yield useful results--including,perhaps,a definition of 'manipulative capacity of a package'(with associated 'partial capacities');and, ultimately,an axiomatization of this whole area of research.If intrinsic,localized properties are to be defined meaningfully, one must suppose that *near-optimal use* is made,of algorithms which are,themselves,*near-optimal for a given(class of) data/calculation pair(s)*and,the first step in this direction,is,to try to specify precisely the(operational) meanings of the italicized phrases.

Of more immediate practical significance are indications of the *relative* efficiency of,say,systems,S_1 and S_2.A general representation of this parameter could have the form: $e^* := F(p_1^1,\ldots,p_m^1;p_1^2,\ldots,p_n^2;C)$.Here,the superscripts label the two systems, each of which has its own 'unit-operation parameters',p_k^j ,and,the symbol C stands for an adequate specification of the computation(as a calculation/data pair).The form of dependence of F on C could be suggested by breaking C into its 'hierarchical components',and then,constructing F from information about these components.Of course,all of this is very vague;but it could be made precise,in a concrete situation,without undue difficulty,provided that sufficiently detailed algorithms for the components of C were available.The dependence of F on the p_k^j would be determined,almost automatically by the algebraic/analytical procedures required to complete the computation,C.This idea,like the possible definitions of intrinsic, characteristic parameters,is worthy of close investigation. For instance,could one derive estimates for relative efficiency in complex calculations,<u>in general</u>,from similar estimates for basic calculations--without descending to the level of the most elementary components of C(as envisaged above,as a possible mode of determination of F)?These,and analogous,questions,raise many interesting problems,but these will not be pursued here.

The most direct measurement of relative performance is obtained by means of <u>timing</u> comparisons.Unfortunately,a moment's reflection shows that the apparent simplicity (and,adequacy)of this approach,is illusory,since,it is virtually impossible to ensure that <u>exactly the same calculation</u> is done,on all machines involved in the comparison.As a rough,practical guide it isuseful:certain types of calculations seem to be 'inherently ill-fitted for execution by some computation packages;and, relatively'well-suited' for others.Although a deeper analysis might reveal the source(s) of this phenomenon,it is unlikely that much could be done to produce a greater uniformity of performance among various systems,since,frequently,the difficulties are rooted in fundamental features of the language/machine designs.Consequently,as symbolic computation becomes more widespread(as it is likely to do,with the development of faster,cheaper,microprocessors--see,e.g.,Miola(1975),Rich and Stoutemyer(1979),for examples of symbolic computations using <u>minicomputers</u>) the assessment of 'worst-case cost' will assume critical importance.Moreover,in line with the above remarks on the possibility of estimating efficiencies for complex computations,from results obtained in relatively simple cases,one would hope to be able to achieve this end in the special case of direct,timing comparisons—since, such a procedure would be of wide application.An idea of how timings can differ, from system to system,is given by the 'league table' compiled by d'Inverno(1978), for a relativity calculation.A useful,general,'snapshot comparison' of several systems(as of about 1974) is provided by the table(due,mainly,to Sundblad)from Cohen et al.(1976);both of these tables are reproduced,here.

17.

The principal thesis of this paper may be summarised(indeed,'sloganized'!) as follows: '<u>sufficiently uniform combinations of (finitely) many (uniformly) effective approximation procedures,are,themselves,(uniformly) effective</u>'. Like all slogans,this one is not intended to be taken literally;nor,to be subjected to intensive logico/mathematical investigation--but,it does convey,<u>in essence</u>,how symbolic computation packages can be used,creatively,by mathematicians,in all fields.The crucial matter is the interpretation of the terms 'uniform',and,'effective';and,many different interpretations have been given in the course of this paper—depending on the field within which approximations were being sought.A prerequisite for the use of what I have called 'symbolic analysis',is the formulation of all problems in a constructive manner(however apparently abstract they may seem to be).With this precondition,the ambitious mathematical schemes outlined in Section 15(as well as many others)are quite feasible,at the level of sophistication now attained.Their detailed implementations constitute major research projects;but, my objective has been,almost entirely,to suggest ways in which significant analytical procedures can be <u>developed</u>,and then,implemented,<u>approximately</u> ,with error estimates at each stage of the computation (often,in cases where the calculations would be prohibitively complicated without a computer).Moreover,the speed of machine calculation is such,that variants of conventional methods may be tried, experimentally,and,'partial proofs' may be explored,heuristically.The large number of references cited here,reflects the great variety of fields considered,and,the diversity of 'almost constructive methods' already in existence(but,mostly,neglected,because of their intractability).Section 15 contains merely a small selection of the vast array of effective procedures that could be assembled for use in symbolic analysis.The crucial breakthrough,allowing 'symbolic manipulation'(as described,e.g.,in Sammet(1967,1969),Tobey(1971)) to be transformed into 'symbolic analysis'(as exhibited,already,in many of the papers given at the MACSYMA Users' Conferences(Fateman(1977), (1979)),and the EUROSAM '79 meeting(Ng(1979)) --to give only the most obvious examples) resided in the fundamental improvements made in general algorithms for:(a)factorization,over algebraic number fields;(b) other calculations involving algebraic number fields,including the determination of multivariable greatest common divisors;and,(c)formal integration,of transcendental and algebraic functions of one variable.These advances are associated,especially,with Collins(1967),Zassenhaus(1969),Brown(1971),Wang(1971),Moses/Yun(1973), Miola/Yun(1974),Wang/Rothschild(1975),and Wang(1976), and,a collection of algorithms underlying the <u>SAC</u>-1 system(see,Collins(1971))-- for items (a) and (b).The remarkable progress in algorithmic integration stems from Ritt's(1948) presentation, and extension,of Liouville's(18333-1840) results;and,the formulation of integration problems in terms of differential fields(the basic formalism,and theorems,being, again,due to Ritt(1950)).Thereafter,the main theoretical contributions came from Risch(1969,1970),for 'transcendental' integrands,and,from Davenport(1979a,b,c),for 'algebraic' integrands.(See,Section 12,for more details of topics (a),(b) and (c)). To make these mathematical advances practically useful,the fundamental algorithms must be implemented effectively—and extremely nontrivial task,which is still to be completed,though,much has been achieved in both hardware and software improvements.Indeed,the design groups of all the systems discussed in this paper are engaged,regularly,in projects to incorporate in their systems the most efficient procedures available(and compatible with their basic design aims).For this reason, it is essential to check,with members of the design groups,all technical information about the current state of(and,implementations of) systems.

On the mathematical side,what has emerged very strongly,in the past decade,is the pervasiveness of algebraic geometry(as a means of unifying procedures involving fractional power series,as a basis for finite element calculations,for domains with curved boundaries;and,above all,as the key to constructing a decision procedure for the integration of algebraic functions).This is,perhaps,not so surprising,after all,if one reflects that algebraic geometry is concerned,mainly,with the systematic study of rings of (formal) power series,and,of varieties(intersections of zero-sets of finite numbers of (homogeneous) n-variable polynomials—over some 'ground field').A glance at textbooks on classical algebraic geometry(e.g.,Lefschetz(1953)) makes the connection with effective methods in symbolic analysis very clear.Nevertheless,the conversion of semi-qualitative procedures into applicable algorithms has been accomplished,so far,only in a few types of calculation. This 'conversion' constitutes a most important aim,deserving close attention. Whilst I am aware that the generality,and scope,of the schemes I have proposed in this paper(and,of the abstract approach they embody)is unusual in the context of computing,it does represent an effor to reduce the emphasis on purely numerical routines,which have dominated numerical analysis for so long.If rigorous,effective, (symbolic) procedures(with their concomitant error estimates) are available,then, they should be used,since,the information they contain,about the analytical properties of functions,cannot be matched,even remotely,by purely numerical output.It is interesting to note that,over ten years ago,Engeli(1969),using his crystal ball to look a decade ahead,envisaged a 'general mathematical utility',in which collaboration between mathematicians and computers would extend(in principle) to all areas of research.The current state of symbolic computation is such,that this vision is no longer futuristic.

For all of the topics discussed in Section 15,I have tried to find existing treatments where the basic aim was to produce effectively calculable results;or,failing this,where,the theorems are presented in a manner allowing the development of 'approximate realizations'.I believe that a very strong case has been made,here, for the extensive study of symbolic analysis;and,that the material given in Sections 14 and 15 represents an essential 'first step' in this direction.Hopefully,it will be of sufficient interest to mathematicians in all areas,for them to implement procedures pertaining to their fields of special concern.

In case it should be thought that my appreciation of mathematics is confined solely to algorithmic procedures,may I declare,in conclusion,my admiration for all who create beauty in mathematics--however esoterically!—though,that beauty is enhanced,for me,in proportion as it unifies apparently disparate fields,and forms a basis for effective calculation.Disinterestedly abstract mathematical theories are,almost always,necessary precursors to constructive procedures.It is,by no means,my intention to detract from the unique blend of abstraction,economy and elegance,which characterizes the greatest contributions to axiomatic mathematics. On the contrary,it is my principal aim to show that,not only are they aesthetically satisfying--most of them are,also,of considerable(and,often,unexpected) practical use.Although many people may share these views,they are stated only very rarely in print--and,seldom reflected in the contents of books at graduate level, or,in research papers.Because of this situation,I have emphasized the potential use of symbolic analysis in predominantly 'theoretical',as well as 'applied' contexts.

Acknowledgements.
Part of my work on this paper was supported by the Polytechnic of Central London. I am grateful to Dr A.C.Norman,of the University of Cambridge Computer Laboratory,for preliminary information,and references on symbolic computation packages; and,to various people who sent me papers and manuals.

SYSTEM	SOURCE	MACHINE	WORD LENGTH IN BITS	CYCLE TIME IN MICRO-SECS	MULTIPLICATION TIME IN MICRO-SECS	COMPUTATION TIME IN SECS	MEMORY REQUIRED IN K-WORDS
GRAD-ASSISTANT	[24]	IBM 7090	36	2.4	34	1,020	32
LAM	[7]	IBM 360/75	32	1.5	2	104	Not available
ALAM	[24]	ATLAS 1	48	3	8	240	50
CLAM	[26]	CDC 6600	60	0.8	1	18	40
SHEEP	[30]	DEC PDP 10	36	1	10	30	40
REDUCE	[5]	DEC PDP 10	36	1	10	360	70
REDUCE	[7]	IBM 360/75	32	1.5	2	856	Not available
CAMAL	[5]	TITAN	48	4	8	140	18
CAMAL† (in coords.)	[75]	IBM 360/75	32	1.5	2	84	93
CAMAL† (null frame)	[75]	IBM 360/75	32	1.5	2	22	68
FORMAC	[24]	IBM 7094	36	2	10	1,800	32
FORMAC	[42]	IBM 360/165	32	1.5	2	50	50
FORMAC	[7]	IBM 360/75	32	1.5	2	162	Not available

Table of comparative statistics for the standard Bondi metric calculation

†The actual quantities computed here are not exactly the same as in the other systems.

(From d'Inverno(1978))

(From Cohen et al.(1976))

Summary of Properties of Algebraic Systems (input/output facilities can be judged from the examples given in the text).

Property	ALTRAN	FORMAC	LAM	REDUCE	SAC-1	SYMBAL
Version	1.9 (1974)	1968	1970	1974	1973	1970
Minimum core memory	260 kbytes (IBM 360)	160 kbytes (IBM 360)	150 kbytes (IBM 360)	300 kbytes (IBM 360)	130 kbytes (IBM 360)	25000 words (CDC6000)
Core memory for problem I	350 kbytes	300 kbytes	400 kbytes	500 kbytes	?	33000 words
Implementation language	FORTRAN	IBM assembler	LISP	LISP	FORTRAN	CDC-assembler
Computer	Any	IBM 360/370	Many	Many	Any	CDC6000
Distribution, addresses in appendix	Free for educational institutions	Free from IBM Program Library	Free	Free	Free	Commercial
Distributed as	Magnetic tape	Magnetic tape	Magnetic tape	Magnetic tape	Punched cards	–
Maintenance	Very good	None	Fair	Good	Fair	–
Correction of system	Easy without complete regeneration	–	Easy but complete regeneration necessary	Easy but complete regeneration necessary	Easy without complete regeneration	–
Dialog version existing	No	Yes, for IBM-360/67 FORDECAL (Grenoble) SYMBAS (Aachen)	No	Yes, for DEC-10	No	No
Syntax	FORTRAN and PL/1-like	FORTRAN or PL/I	LISP	ALGOL-like	FORTRAN (METASAC: ALGOL-like)	Improved ALGOL
Declaration of variables (security)	Everything type and layout	No	No	Some	No (Metasac: everything)	Only spelling
Debugging facilities	Very good	Good	Poor	Fair	Almost none	Good
Output form	Some choice	Some choice	No choice	Good choice	No choice	No choice
User's documentation	Very good manual	Good manual	Good manual	Incomplete manual	Good description	Good manual
System documentation	Very good listing and manual	Not available	Good listing	Good full listing	Very good description and listing	Not available
Exact arithmetic	Fix length	2295 digits	Infinite precision	Infinite precision	Infinite precision	Infinite precision
Floating point arithmetic	Slow	Fast	None	Very slow (LISP)	Fast	None
Most general expression	Rational expression	May include functions	May include functions	May include functions	Rational expression	Rational expression
Knowledge of elementary functions	None	Most	Most	Some	None	None
Definition of differentiation rules	Explicit in procedure	Very good possibilities	Good possibilities (in LISP)	Very good possibilities	Explicit in procedure	Explicit in procedure
Pattern matching	Limited	Very limited	Very limited	Good	None	None
Analysis of expressions	Very good	Fair	Poor	Good	Fair	Good
Rational function algorithms (division, gcd)	Good	None	None	Good	Very good	None
Modernity of algorithms	Good	Bad	Bad	Good	Very good	Bad
Handling of truncated power series	Very good	None	None	Good	None	Fair
Formalism for vectors and matrices	Good	Fair	Good	Fair	None	Very good
Gamma matrix algebra	No	No	No	Yes, built in	No	No
Noncommutative algebra	No	No	No	Under testing	No	No

18. References.

A. Documentation for symbolic computation systems.

BROWN,W.S.,et al.(1971-77): ALTRAN User's Manual(Bell Laboratories,Murray Hill,NJ).

GLUSHKOV,V.M.,et al.,Kibernetika 3,102-134(1971);English translation(1974),by Consultants Bureau,227,W.17th Street,New York,NY 10011; 'ANALITIK (ALGORITHMIC LANGUAGE FOR THE DESCRIPTION OF COMPUTING PROCESSES USING ANALYTICAL TRANSFORMATIONS).

FITCH,J.P.(1975-): CAMAL User's Manual(Computer Laboratory,University of Cambridge).

BAHR,K.A.(1978-) : FORMAC 73 User's Manual(Computer Science Department,Pennsylvania State University).

MATHLABGROUP(1977-):MACSYMA Reference Manual,Version Nine(M.I.T.,Cambridge Mass.).

HEARN,A.C.(1973-):REDUCE User's Manual,2nd Edition(University of Utah).

COLLINS,G.E.,et al.(1970-):SAC-1 Technical Reports(University of Wisconsin,Madiso

COLLINS,G.E.and S.C.SCHALLER(1976):SAC-1 User's Guide(University of Wisconsin).

GRIESMER,J.H.,R.D.JENKS and D.Y.Y.YUN(1975):SCRATCHPAD User's Manual(IBM Watson Research Center,Yorktown Heights,NY).

——— , ——— and ——— (1976):A Set of SCRATCHPAD Examples(IBM Watson Research Center,Yorktown Heights,NY).

ENGELI,M.(1969):'Formula Manipulation--The User's Point Of View',Adv. in Info. Systems Sci.,1,117-171.

——— (1975):'An Enhanced SYMBAL System',SIGSAM Bull.9(4),21-29(1975).

——— (1975-):SYMBAL Manual(FIDES Trust Company,Zurich,Switzerland)

B. General references.

AHLBERG,J.H.,E.N. NILSON and J.L.WALSH(1967):'The Theory of Splines and Their Applications'(Academic Press,New York)

ALLEN,J.(19):'Anatomy of LISP' (Prentice Hall).

ALLGOWER,E. and K.GEORG(1980):'Simplicial and Continuation Methods for Approximating Fixed Points and Solutions to Systems of Equations',SIAM Review 22 No.1,28-85.

ANDERSON,C.M. and A.K.NOOR(1977),in 'Proc.of the 1977 MACSYMA Users'Conference', (NASA Report CP-2012),161-175.

APPEL,K and W.HAKEN(1977a):'Every planar map is four colorable'.Part I:Discharging', Illinois J.Math. 21 ,429-490.

APPEL,K.,W.HAKEN and J.KOCH(1977b):'Every planar map is four colorable.Part II: Reducibility',Illinois J.Math.21,491-567.

ARNOL'D,V.I.(1971):'On matrices depending on parameters'.Russian Math.Surveys 26 ,29-43.

AVERBUH,V.I.and O.G.SMOLJANOV(1967):Russian Math.Surveys 22(6),201-258.

—— and —— (1968): —— 23(4),67-113.

AYOUB,R(1963):'Introduction to the Analytic Theory of Numbers',Mathematical Surveys,No.10(AMS,Providence,RI).

AZIZ,A.K.(ed.)(1972):'Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations'(Academic Press).

BARTON,D.and J.P.FITCH(1972a):in,Reports on Progress in Physics 35,235-314.

—— and —— (1972b):'A Review of Algebraic Manipulative Programs and Their Application',The Computer J.,Vol.15(4),362-381.

BAKER, A.(1974):'Transcendental Number Theory'(Cambridge University Press).

BAKER,J.A.and J.L.GAMMEL(1970):'The Padé Approximant in Theoretical Physics', (Academic Press,New York).

BALINSKI,M.L.(1974):'Pivoting and Extensions',Mathematical Programming Study No.1, (North-Holland/Elsevier).

BARBER,M.(1977):Physics Reports 29

BATEMAN,H.(1932):'Partial Differential Equations'(Cambridge U.Press;reprinted,Dover).

BAXTER,R.J.(1971a):'Partition Function for the Eight-Vertex Model',Ann.Phys.

—— (1971b):'One-dimensional Anisotropic Heisenberg Chain',Ann.Phys.

BECKENBACH,E.F.(ed.)(1952):'Construction and Application of Conformal Maps' (National Bureau of Standards Series,No.18,Washington DC).

BERENSTEIN,C.A.and M.A.DOSTAL(1972):'Analytically Uniform Spaces and their Applications to Convolution Equations'(Springer,LNM 256)

BERGMAN,S.(1950):'The Kernel Function and Conformal Mapping'(AMS Math.Surveys,No.5).

BELTRAMI,E.(1970):'An Algorithmic Approach to Nonlinear Optimization'(Academic Press)

BERSTRÖM,H.(1963):'Limit Theorems for Convolutions'(Wiley/Almqvist and Wiksell).

BERKELEY,E.and D.BOBROW,(eds.)(1965):'The programming language LISP;its operation and applications',(Clearinghouse of the U.S.Dept.Commerce).

BETH,E.R.(1968):'The Foundations of Mathematics',2nd Edition(North-Holland).

BIEBERBACH,L.(1952):'Conformal Mapping',(Chelsea Publishing Co.,New York).

BISHOP,E.(1967):'Foundations of Constructive Analysis'(McGraw Hill).

BLISS,G.A.(1966):'Algebraic Functions'(AMS Colloquium Publication;reprinted,Dover).

BIRKHOFF,G.(1973):'Lattice Theory',3rd Edition(AMS Colloquium Publication,No.25).

BONGAARTS,P.J.M. and Th.J.SISKENS(1974): 'Observables,Constants of the motion and ergodicity in quantum-statistical mech.of finite systems',Physica 71,529-559.

BRINCH HANSEN,P.(1975):'The programming language Concurrent Pascal',IEEE Trans. Software Eng.1(2),199-207.

BREBBIA,C. and A.C.WALKER(1980):'Boundary Element Techniques in Engrg.'(Butterworth).

BROWN,W.S.(1971):'On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors',JACM,Vol.18,478-504.

BROWN,W.S. and A.C.HEARN(1978):'Applications of Symbolic Algebraic Computation', Report UCP-61(University of Utah Computl.Phys.Group).

BRIDGES,D.S.(1979):'Constructive Functional Analysis'(Pitman Res.Notes in Math.,28).

BUREAU,F.(1955):'Divergent Integrals and Partial Differential Equations', Commun.Pure and Applied Math.Vol.VIII,143-202.

BROWN,K.Q.(1979):'Geometric Transforms for Fast Geometric Algorithms', Report CMU-CS-80-101(Carnegie-Mellon University).

BRENT,R. and H.T.KUNG(1978):'Fast Algorithms for Manipulating Formal Power Series', JACM,Vol.25(4),581-595.

BURKS,A.W.,D.W.WARREN and J.B.WRIGHT(1954):'An analysis of a logical machine using parenthesis-free notation',Math.Tables and Other Aids Compn.,8.

CAMERON,R.H.(1951): 'A Simpson rule for the numerical evaluation of Wiener integrals in function space',Duke Math.J.18(1),111-130.

—— (1954):'The general heat flow equation and a corresponding Poisson formula',Ann.Math.59(3),434-461.

—— and W.T.MARTIN(1944):'Tranformations of Wiener integrals under translations', Ann.Math.,45(2),386-396.

—— and —— (1945a):'Transformations of Wiener integrals under a general class of linear tranformations',Trans.AMS,58(2),184-219.

—— and —— (1945b):'Evaluation of various Wiener integrals by use of certain Sturm-Liouville differential equations',Bull.AMS,51(2),73-90.

—— and —— (1949):'Transformations of Wiener integrals by nonlinear transformations',Trans.AMS,66(2),253-283.

CAMPBELL,J.A.(1974):'Symbolic computing and its relationship to particle physics', Acta Physica Austriaca,Suppl.XIII,595-647.

CAMPBELL,S.L. and C.D.MEYER(1980):'Generalized inverses of linear transformations', (Pitman).

CANNON,J.J.(1973):'A general purpose group theory program',Proc.Second Internat. Conf.Theory of Groups,Canberra,204-217.

—— (1976a):'The CAYLEY Library of Built-in Functions',Technical Report No.13,Dept.of Pure Mathematics,University of Sydney.

—— (1978):'The group theory language CAYLEY:student reference manual', Department of Pure Mathematics,University of Sydney.

—— (1976b):'A draft description of the group theory language CAYLEY', Technical Report No.12,Dept.of Pure Mathematics,U.of Sydney.

CAVINESS,B.F.(1970):'On canonical forms and simplification',JACM,<u>17</u>(2),385-396.

—— and R.FATEMAN(1976):'Simplification of radical expressions',Proc.ACM Symp.Symbolic and Algebraic Computation,329-338.

CHOW,S.-N.,HALE,J.K. and MALLET-PARET,J.(1975a):'Applications of generic bifurcation,I,Arch.Rat.Mech.Anal,<u>59</u>,159-188.

—— , —— and —— (1975b):'Applications of generic bifurcation,II,Arch.Rat.Mech.Anal.<u>62</u>,209-236.

CIARLET,P.G.(1977):'The finite element method for elliptic problems'(North-Holland).

COATES,J.(1970):'Construction of rational functions on a curve',Proc.Cam.Phil. Soc.,<u>68</u>,105-123.

COHEN,E.G.D.(ed.)(1962):'Fundamental problems in statistical mechanics',North-Holland.

—— (ed.) —— —— —— II,North-Holland.

COHEN,H.I.,O.LERINGE and Y.SUNDBLAD(1976):'The use of algebraic computing in general relativity',Gen.Relativity and Gravitn.<u>7</u>(3),269-286.

COHEN,P.J.(1963):'The independence of the Continuum Hypothesis,I',Proc.National Acad.Sci.<u>50</u>,1143-1148.II.Proc.Natl.Acad.Sci.,<u>51</u>,105-110(1964).

COLLINS,G.E.(1957):'Tarski's decision method for elementary algebra',Proc.Summer Inst.Symbolic Logic,Cornell University,p64.

—— (1966):'PM,a system for polynomial manipulation',CACM,<u>9</u>(8),578-589.

—— (1967):'Subresultants and reduced polynomial remainder sequences', JACM,<u>14</u>(1),128-142.

—— (1971):'The calculation of multivariate polynomial resultants', JACM,<u>18</u>(Oct.,1971),515-532.

COLTON,D.(1976a):'Partial differential equations in the complex domain',Pitman Research Notes in Mathematics,No.4.

—— (1976b):'Solution of boundary value problems by the method of integral operators',Pitman Research Notes in Mathematics,No.6.

COOKE,R.G.(1950):'Infinite matrices and sequence spaces'(Macmillan;repr.Dover,1966).

COOPERSMITH,M.H.(1968):'Analytic free energy:a basis for scaling laws',Phys. Rev. ,230-240.

COURANT,R.and ROBBINS,H.(1969):'What is mathematics?'(Oxford University Press).

DAVENPORT,J.H.(1979a):Ph D Thesis(University of Cambridge).

—— (1979b):'The computerisation of algebraic geometry',Proc.'EUROSAM '79', Lec.Notes Comp.Sci. No.72,E.W.Ng(ed.),119-133.

—— (1979c):'Algorithms for the integration of algebraic functions', Proc.'EUROSAM '79',LNCS No.72,415-425.

—— (1980):'Anatomy of an integral',ACM SIGSAM Bulletin.

DAVIES,E.B.(1976):'Quantum theory of open systems',(Academic Press).

DAVIS,M.(1957):'Computability and unsolvability',(McGraw Hill).

—— (1977):'Applied nonstandard analysis',(Wiley).

DAVIS,P.(1963):'Interpolation and approximation',(Blaisdell;repr.,Dover).

DUNLOP,F.and C.M.NEWMAN(1975):'Multicomponent field theories and classical rotators',Commun.Math.Physics,44,223-235.

EDWARDS,R.E.(1965):'Functional analysis—theory and applications', (Holt,Rinehart and Winston).

EDWARDS,S.F.and A.LENARD(1962):'Exact statistical mechanics of a one-dimensional with Coulomb forces',II.The method of functional integration, J.Math.Phys.,Volume 3,No.4,778-792.

EHRENPREIS,L.(1970):'Fourier analysis in several complex variables', (Wiley/Interscience).

EISENHART,L.P.(1927):'Non-Riemannian geometry',(AMS Colloqu.Publ.No.8;repr.,1968).

ELVEY,J.S.N.(1973):'On the Yang-Lee distribution for a van der Waals fluid near the critical point',Progr.Theoretical Phys.,49,1428-1439.

—— (1974):'The Yang-Lee distribution for a class of lattice gases', Commun.Math.Physics,35,101-112.

—— (1978):'Constructive methods for boundary value problems,with emphasis on plane elastostatics and formula-manipulative computing', (unpublished report,pp154,supported,in part,by the Sci.Res.Counci

ENGELI,M.(1966):'Design and implementation of an algebraic processor', Habilitationsschrift,ETH,Zurich,Switzerland.

ENGLAND,A.(1971):'Complex variable methods in elasticity',(Wiley).

EPSTEIN,H.I.(1975):'Algorithms for Elementary transcendental function arithmetic', Ph D Thesis,University of Wisconsin,Madison.

ERDELYI,A.,MAGNUS,W.,OBERHETTINGER,F.and F.G.TRICOMI(1954):'Tables of integral transforms,Volumes I,II.''Higher transcendental functions',Vols.I,II,III'.

ERDOS,P.and S.ULAM(1971):'Some probabilistic remarks on Fermat's Last Theorem', Rocky Mountain J.Math.,Volume 1(4),613-616.

FELSCH,V.(1977-):(Computer-based biblio.on symbolic computation in group theory,etc.) University of Aachen.

FAIRWEATHER,G.(1978):'Finite element and Galerkin methods for differential equations',Lecture Notes in Pure and Applied Math.(Dekker).

FATEMAN,R.J.(1972):'Essays in algebraic simplification',Project MAC,Report MAC TR-95.

—— (1977):'Some comments on series solutions',Proc.1977 MACSYMA Users' Conference,(Report NASA CP-2012),327-346.

FARQUHAR,I.E.(1964):'Ergodic theory in statistical mechanics'(Wiley/Interscience).

FELLER,W.(1957):'An introduction to probability theory and its applications, Volume 1', 2nd Edition,(Wiley).

—— (1966):'An introduction to probability theory and its applications, Volume 2, (Wiley).

FINK,A.M.(1974):'Almost periodic differential equations',(Springer LNM,No.377).

FISHER,M.E.(1964):'The free energy of a macroscopic system',Arch.Rat.Mech. Anal.Volume 17,377-410.

FITCH,J.P.(1979):'The application of symbolic algebra to physics—a case of creeping flow',Proc.'EUROSAM '79',E.W.Ng(ed.),(LNCS,72),30-41.

FORSYTH,A.R.(1918):'Theory of functions',3rd Edition,(Cambridge U.P.;repr.,Dover).

FRICK,I.(1977):'The computer algebra system SHEEP,what it can and cannot do in general relativity'(Univ.Stockholm Inst.Physics,Report 77-14).

FRISCH,H.and J.L.LEBOWITZ(1964):'The equilibrium theory of classical fluids', a Lecture Note and Reprint Volume,(Benjamin,New York).

GAIER,D.(1964):'Konstruktive methoden der konforme abbildung',(Springer Tracts in Natural Philosophy,No.3).

—— (1976):'Intergralgleichungen erster Art und konforme Abbildung', Math.Zeit.147,113-129.

GAIER,D.(1979):'Das logarithmische Potential und die konforme Abbildung mehrfach zusammenhangender Gebiete'(Preprint,University of Giessen).

GARNETT,J.(1972):'Analytic capacity and measure',(Springer LNM,No.297).

GEDDES,K.O.(1977):'Symbolic computation of recurrence equations for the Chebyshev series solution of linear ODE's',Proc.1977 MACSYMA Users' Conference,(NASA report CP-2012),405-423.

——— (1978):'An ALTRAN implementation of the fraction-free Padé algorithm', Research report CS-78-10(University of Waterloo,Ontario,Canada).

——— (1979):'Convergence behaviour of the Newton iteration for first-order differential equations',Proc.'EUROSAM '79',(Springer LNCS,72), 188-199.

GEL'FAND,I.M.and A.M.YAGLOM(1960):'Integration in functional spaces and its applications in quantum physics',J.Math.Phys.,Volume 1(1),48-69.

——— et al.(1964-1968):'Generalized functions,Volumes I-V',(Academic Press).

GLUSHKOV,V.M. et al.(1978):(A description of the latest version of ANALITIK), Kibernetika,Volume 5 (in Russian).

GOLUBITSKY,M.and D.SCHAEFFER(1979):'Imperfect bifurcation via singularity theory', Commun.Pure and Appl.Math.XXXII,21-98.

——— and V.GUILLEMIN(1973):'Stable mappings and their singularities', (Springer Graduate Texts in Mathematics,No.14).

GOLUZIN,G.M.(1969):'Geometric theory of functions of a complex variable', AMS Translations of Mathematical Monographs,Volume 26.

——— and V.I.KRYLOV(1933):'A generalization of Carleman's formula and its application to analytic continuation of functions', Mat.Sbornik,40,144-149.

GONZALEZ,M.J.and C.V.RAMAMOORTHY(1970):'Program suitability for parallel processing', in,Hobbs,L.C.et al.(eds.),'Parallel Processor Systems,Technologies and applications,(Spartan Books,New York,1970).

GOSPER,R.W.(1976):'A calculus of series rearrangements',pp 121-151,in Traub(1976), (ed.)'Algorithms and Complexity,New Directions and Recent Results'(Academic Press).

——— (1977):'Indefinite hypergeometric sums in MACSYMA',Proc.1977 MACSYMA Users' Conference(NASA report CP-2012),237-251.

GRAUERT,H.and K.FRITZSCHE(1976):'Several complex variables',(Springer,GTM,No.38).

GRAY,A.and L.Vanhecke(1979):'Riemannian geometry as determined by the volumes of small geodesic balls',Acta Math.142,157-198.

GREEN,A.E. and W.ZERNA(1968):'Theoretical elasticity',2nd Edition,(Oxford U.Press).

GRIESMER,J. and JENKS,R.(1971):(A description of the SCRATCHPAD system),in,Proc. Second Symp.Symbolic and Alg.Manip.(S.R.Petrick,ed.);also available as IBM Research Report RC 3925(IBM Watson Res.Center,Yorktown Heights).

HARRINGTON,S.(1977):'A new symbolic integration system in REDUCE',(University of Utah Symbolic Computation Group,Report UCP-57.

HAKEN,W.(1978):'Combinatorial aspects of some mathematical problems',Proc.Internat. Congress Math.(O.Lehto,ed.),Volume 2,953-961.

HARDY,G.H.(1949):'Divergent series',Oxford University Press.

HADAMARD,J.(1903):'Lecons sur la propogation des ondes...'(Hermann;repr.Chelsea,1949).

——— (1932):'Le Probleme de Cauchy ...',(Hermann;repr.,Dover).

HARDY,G.H. and J.E.LITTLEWOOD(1920):'Some problems of 'Partitio Numerorum':I. A new solution of Waring's Problem',Nachr.K.Gesell.Wissen zu Gottingen, Math-phys.Klasse,33-54.(and other papers in the 'Partitio Numerorum'set).

HEARN,A.C.(1971): Commun.ACM 14,511-516.

——— (1977a):'Algebraic manipulation by computer',Report UUCS77-115 (U.of Utah).

——— (1977b):'The structure of algebraic computations',in,Proc.Fourth Colloq. Adv.Comp.Methods in Theor.Phys.(St Maxime,France).

HERMES,H.(1969):'Enumerability,computability,decidability',(Springer,'Grundl.',127).

HELLER,D.(1978):'A survey of parallel algorithms in numerical linear algebra', SIAM Review,Volume 20(4),740-777.

HENRICI,P.(1974):'Applied and computational complex analysis,Vol.I',(Wiley).

——— (1977):' ——— ——— ——— Vol.II',(Wiley)

——— (1979):'Fast Fourier methods in computational complex analysis', SIAM Review,Volume 21(4),481-527.

HIGGINS,J.R.(1977):'Completeness and basis properties of sets of Special Functions', Cambridge Tracts in Mathematics,No.72).

HILLE,E. and R.PHILLIPS(1957):'Functional analysis and semigroups',AMS Colloquium Publications,No.31.

HILL,T.L.(1956):'Statistical mechanics',(McGraw-Hill).

HINTIKKA,J. and P.SUPPES,(eds.)(1966):'Aspects of inductive logic',(North-Holland).

HEWITT,E. and K.ROSS(1963):'Abstract harmonic analysis,Volume 1',(Springer).

HOARE,C.A.R.(1978a):'A model for communicating sequential processes' (Oxford U.Programming Res.Group,45,Banbury Road,Oxford).

HOARE,C.A.R.(1978b):'Communicating sequential processes',Commun.ACM,Vol.21,666-677.

------ and R.M.McKEAG(1977):'Structure of an operating system',Preprint, Oxford University Programming Research Group,45,Banbury Road,Oxford.

HORNFELDT,L.(1979):'A system for automatic generation of tensor algorithms and indicial tensor calculus,including substitution of sums', Proc.'EUROSAM '79(Springer LNCS No.72),279-290.

HOLMES,P.(1979):'A nonlinear oscillator with a strange attractor',Math.Proc. Royal Society,Volume 292 A,419-428.

HOROWITZ,E.(1969):'Algorithms for symbolic integration of rational functions', Ph.D.dissertation,University of Wisconsin,Madison.

------ (1971):'Algorithms for partial fraction decomposition and rational function integration',in,'SYMSAM 71',441-457.

HOROWITZ,G.(1966):(Asymptotic representations of thermodynamic functions for finite systems),J.Math.Phys.,Volume 7,2261.

------ (1968):(Asymptotic representations of thermodynamic functions for finite systems),J.Math.Phys.

van HOVE,L.(1953):'The occurrence of singularities in the elastic frequency distribution of a crystal',Phys.Rev.,89(6),1189-1193.

HUSBERG,N.(1977):'The English ANALITIK system for MIR-2',Research Report No.3, Helsinki University of Technology,Computing Center.

HUTTON,B.(1977):'Uniformities on fuzzy topological spaces',J.Math.Anal.Applications,Volume 58(3),559-571.

------ and I.REILLY(1980):'Separation axioms in fuzzy topological spaces', Fuzzy Sets and Systems,Volume 3,93-104.

IDEL'SON,A.V.(1964):'Calculi of constructive logic with subordinate variables', Amer.Math.Soc.Translations,Series 2,Volume 99,83-227.

IKEDA,K.(1974a):'On the Yang-Lee distribution of zeros for a gas obeying van der Waals' equation of state,I',Prog.Theor.Phys.52,p 54.

------ (1974b): ------ ------ II.Prog.Theor.Phys.52,p 415.

------ (1974c): ------ ------ III.Prog.Theor.Phys.52,p 840.

------ (1975): ------ ------ IV. Prog.Theor.Phys.53,p 66.

d'INVERNO,R.A.(1969): Computer J.,Volume 12,p 124.

------ (1978):'A review of algebraic computing in general relativity', (U.of Southampton Faculty Math.Studies,Preprint Series,No.11).

------ and R.A.RUSSELL-CLARK(1971):Computer J.,Volume 17,p 229.

ISAACSON,E. and H.B.KELLER(1966):'Analysis of numerical methods',(Wiley).

IWATA,G.(1964):'A method for evaluating a class of partition functions,I', Prog.Theor.Phys.,Volume 31(1),67-82.

JONES,J.P.et al.(1976):'Diophantine reprn.of the set of prime numbers', Amer.Math.Monthly,83,449-464.

JASWON,M.A. and G.SYMM(1977):'Integral equation methods in applied mathematics', (Academic Press).

JAZWINSKI,A.(1970):'Stochastic processes and filtering theory',(Academic Press).

JEFFREYS,H.(1957):'Scientific inference',(Cambridge University Press).

——— (1961):'Theory of probability',3rd Edition,(Oxford University Press).

JENKS,R.D.(1974):'The SCRATCHPAD language',SIGSAM Bulletin,Volume 8(2).

——— (1979):'MODLISP:an introduction',Proc.'EUROSAM '79'(Springer LNCS 72, 467-480).

JONES,C.(1977):Ph.D.Thesis,(Oxford U.Computer Lab.,Programming Research Group).

JONES,G.L.(1966):'Complex temperatures and phase transitions',J.Math.Phys.7,2000-2005

KARP,R.(ed.)(1974):'Complexity of computation',(SIAM-AMS Proc.,Prov.R.I.).

KAC,M.(1959):'Probability and related topics in the physical sciences', (Wiley/Interscience).

KAHANE,J-P.(1968):'Some random series of functions',(Heath Mathematical Monographs).

KANTOROVICH,L.V. and G.P.AKILOV(1964):'Functional analysis in normed spaces', (Pergamon Press).

——— and V.I.KRYLOV(1958):'Approximate methods of higher analysis', (Noordhoff).

KAHRIMANIAN,H.G.(1953):'Analytical differentiation by a digital computer' M.A.Dissertation,Temple University,Philadelphia,Pa.

KARLIN,S.(1959a):'Mathematical methods and theory in games,programming and economics ,Volume 1',(Addison-Wesley).

——— (1959b): ——— Volume 2, (Addison-Wesley).

KATO,T.(1976):'Perturbation theory of linear operators',2nd Edition,(Springer).

KATSURA,S.(1963):'Singularities in first-order phase transitions',Adv.Phys., Volume 12,391-420.

KENDALL,M.G. and A.STUART(1958):'The advanced theory of statistics,Vol.1,(Griffin).

——— and ——— ——— Vol.2,(Griffin).

KIM,W.H. and R.T.-W.CHIEN(1962):'Topological analysis and synthesis of communication networks',(Columbia University Press).

KNOPS,R.J.(ed.)(1972):'Symposium on non-well-posed problems and logarithmic convexity',(Springer LNM,No.316).

—— (1977):(Heriot-Watt symposium on nonlinear analysis), Pitman Research Notes in Mathematics.

KNUTH,D.E.(1969):'The art of computer programming.Volume 2:semi-numerical algorithms',(Addison-Wesley)

KNOPP,K.(1928):'Theory and application of infinite series',(Blackie;repr.Springer).

KOBRINSKII,N.E. and B.A.TRAKHTENBROT(1965):'Introduction to the theory of finite automata',(Studies in Logic:North-Hollanc

KONDO,K.(ed.)(1958):'The RAAG memoirs,Volume 1'(Research Association for Applied Geometry,Gakujutsu Bunken Fukyu-kai,T.I.T.,Oh-okayama meguro-ku, Tokyo,Japan.Also:Dept.of Engineering,University of Tokyo).

—— (1962): ———— Volume 2 ——

—— (1968): ———— Volume 3 ——

KORPELA,J.(1976):'General characteristics of the ANALITIK language', ACM SIGSAM Bulletin Volume <u>10</u>(3),29-48.

—— (1977):'On the MIR series of computers and their utilisation for analytical calculations',Proc.Fourth Internat.Colloq.Adv.Comp. Methods in Theor.Physics,(St Maxime,France).

KRASNOSEL'SKII,M.A.(1964):'Topological methods in the theory of nonlinear integral equations',(Pergamon Press).

—— et al. (1966):'Plane vector fields',(Illiffe).

—— et al. (1972):'Approximate solution of operator equations',(Noordhoff).

KRYLOV,V.I.(1962):'Approximate calculation of integrals',(Macmillan,New York).

KUIPERS,B.(1973):(Project on MACSYMA solution of ODE's,M.I.T.).

KUNG,H.T. and J.F.TRAUB(1978):'All algebraic functions can be computed fast', J.ACM,Volume <u>25</u>(2),245-260.

KUPRADZE,V.D.(1965):'Potential theoretic methods in elasticity',(Pergamon Press).

—— et al.(1979):'Three-dimensional problems of elasticity and thermoelasticity' (North-Holland).

LAFFERTY,E.L.(1977):'Power series solutions of ordinary differential equations in MACSYMA', Proc.1977 MACSYMA Users' Conference, NASA Report CP-2012,347-360.

LATTES, .and J-L.LIONS(1969):'The method of quasi-reversibility...',(Elsevier).

LEFSCHETZ,S.(1953):'Algebraic geometry',(Princeton U.P./Oxford U.P.).

LEE,T.D.and C.-N.YANG(1952):'Statistical theory of equations of state and phase transitions.II.Lattice gas and Ising model',Phys.Rev.87,410-419.

LEVINSON,N.(1974):'More than one third of the zeros of Riemann's Zeta Function lie on $\sigma = 1/2$ ',Adv.Math.Volume 13,383-486.

—— and R.Redheffer(1970):'Complex variables',(Holden-Day).

LEKHNITSKII,S.G.(1963):'Theory of elasticity of an anisotropic elastic body', (Holden-Day).

LIOUVILLE,J.(1833):J.Ecole Polytech.XIV,Sec.23,124-193.

—— (1835):J.f.d.reine u.angewandte math.,XIII,93-118.

—— (1837):J.Math.Pures et appl.II,56-104;III,523-546.

—— (1839):—— IV,423-456.

—— (1840):—— V,34-36;441-464.

LIPSON,J.D.(19):'Chinese Remainder and interpolation algorithms',

LIVERMAN,T.(1964):'Generalized functions and direct operational methods', (Prentice Hall).

LORENZ,E.N.(1963):J.Atmos.Science,Volume 20,130141.

LOWEN,R.(1979):'Fuzzy neighbourhood spaces',Preprint,Math.Dept.,Vrije U.,Brussel(s).

LOOMIS,L.H.(1953):'Abstract harmonic analysis',(van Nostrand).

LOOS,R.(1972):'Analytic treatment of three similar Fredholm equations', ACM SIGSAM Bulletin,Volume 21,32-40.

LUXEMBURG,W.(ed.)(1969):Proc.Internat.Symp.Nonstandard Analysis,(North-Holland).

——(ed.) (1972):Proc.Symp.Oberwolfach,(North-Holland).

—— (1973):'What is nonstandard analysis?',in,'Papers in the foundations of mathematics',Amer.Math.Monthly,80(6),Part II,38-67.

LYAPUNOV,A.M.(1906):P.1.Zap.Akad.Nauk,St Peterburg.

LYUSTERNIK,L.A.and V.I.SOBOLEV(196):'Elements of functional analysis', MACK,C.(1975):'Integration of affine forms over Elementary functions', (University of Utah Computational Physics Group,Report No.UCP-39).

MACKEY,G.W.(1974):'Ergodic theory and its significance for statistical mechanics and probability theory',Advances in Mathematics,12,178-268.

MACLANE,S.(1967):'Homology',(Springer,Grundlehren,band 114).

MANIN,Ju.I.(1958):'Algebraic curves over fields with differentiation', transl.in,AMS Transl.Series 2,No.37,59-78(1964).

—— (1963):'Rational points of algebraic curves over function fields', transl.in,AMS Transl.,Series 2,No.50,1896234(1966).

MANOVE,M et al.(1968):Proc.1966 IFIPS Conference on symbolic manipulation languages,D.G.Bobrow,ed.,(North-Holland),86-102.

MARSDEN,J. and M.McCRACKEN(1976):'The Hopf bifurcation and its applications', (Springer).

MASLOV,V.P.(1976):'Operational methods',(MIR Publications,Moscow).

MATIJASOVIC,Ju.V.(1970):'Enumerable sets are diophantine',Soviet Math.Dokl.,11.

MAURER,W.(1972):'The programmer's Introduction to LISP',(Macdonald/Elsevier).

MAUNDER,C.(1970):'Algebraic topology',

MAZURKIEWICZ, .(1975):

MASON,J.C.(1970): In,'Approximation Theory',A.Talbot,ed.,Academic Press,7-33.

—— (1980):'Recent advances in near-best approximation',Proc.Internat.Conf. Approx.Theory,University of Texas at Austin,January 8-12,1980.

McCARTHY,J. et al.(1965):'LISP 1.5 Programmer's Manual,2nd Edition,(M.I.T.).

McSHANE,E.(1972):'Stochastic differential equations and models of random processes', Proc.Sixth Berkeley Symposium Prob.Statistics,Vol.III,263-294.

MEL'NIKOV,V.K.(1963):'On the stability of the center for time-periodic perturbations',Trans.Moscow Math.Soc.,Volume 12,1-57.

MIKHLIN,S.G.(1957):'Integral equations... ',(Pergamon Press).

MIKHUSINSKI, .(1959):'Operational calculus'.

MILLAR,W.(1978):'Separation of variables... ',(Academic Press).

MILNOR,J.:'Morse theory',(Annals of Mathematical Studies,Princeton U.P).

MIOLA,A.(1974):'The use of the MACSYMA system for solving free boundary problems by the Ritz-Galerkin method'(I.A.C.Report,Series 3,No.9,U.of Rome).

—— (1976a):'Manipolazione algebrica:sviluppo e prospettive', Revista di informatica,vol.VI(1/2),13-23.

MIOLA,A.(1976b):'Sistemi di manipolazione algebrica', Revista di informatica vol.VI(1/2),25-33.

---- and YUN,D.Y.Y.(1974):'The computational aspects of Hensel-type univariate polynomial greatest common divisor algorithms',Proc.'EUROSAM '74', (ACM SIGSAM Bulletin No.31),46-54.

MOISIL,G.(1969):'The algebraic theory of switching circuits',(Pergamon Press).

MOORE,C.N.(1938):'Summable series and convergence factors', AMS Colloquium Publications,No.XXII;reprinted by Dover(1966).

MOORE,R.(1968):'Interval analysis',(Prentice Hall).

---- (1979):'Applications of interval analysis in numerical analysis', (Prentice Hall).

MONTROLL,E.W.(1952):'Markoff chains,Wiener integrals and quantum theory', Commun.Pure Appl.Math.,Vol.5,415-453.

MORSE,M.(1934):'Calculus of variations in the large', AMS Colloquium Publications,No.18;reprinted,1966.

---- (1946):'Topological methods in the theory of functions of a complex variable',Annals of Math.Studies,(Princeton U.P.).

---- and S.S.CAIRNS(1969):'Critical point theory in global analysis and differential topology',(Academic Press).

MORDUKHAI-BOLTOVSKOI,D.(1906-1909):Commun.Soc.Math.Kharkov,X,34-64;231-269.

---- ---- (1913):'On the integration of transcendental functions', Warsaw Univ.Izv.,Nos.6-9 (in Russian).

MOSES,J.(1967):'Symbolic integration',Ph.D.Diss.,M.I.T..

---- (1971a):'Symbolic integration:the stormy decade',CACM,14(8),548-560.

---- (1971b):'Algebraic simplification:a guide for the perplexed',CACM 14,527-537.

---- (1974):'The evolution of algebraic manipulation algorithms', IFIP 74,(North-Holland).

---- and D.Y.Y.YUN(1973):'The EZ GCD Algorithm',Proc.ACM National Convention(8/73).

---- and R.ZIPPEL(1979):'An extension of Liouville's theorem', Proc.'EUROSAM '79'(Springer LNCS No.72),426-430.

MISNER,C.,K.THORNE and J.WHEELER(1973):'Gravitation', (Freeman).

MOUTON,J-P(1979a):'Computer methods for metal-forming problems', Ph.D.Diss.University of Stockholm.

---- (1979b):

MOUTON,J-P. and . ,AMAN(1979):

MUSSER,D.R.(1971):'Algorithms for polynomial factorization', Ph.D.Diss.U.Wisconsin Dept.Comp.Sci.(also,available as tech report 134,Comp.Sci.Dept).

MORREL,B. and J.NAGATA(1978):General Topology and its Applications,9,233-237.

MUSKHELISHVILI,N.I.(1953):'Singular integral equations',(Noordhoff).

——— (1963):'Some basic problems in the mathematical theory of elasticity',3rd Edition,(Noordhoff).

NASHED,M.Z.(1971):'The role of differentials... ',in,'Nonlinear functional analysis'(M.Z.Nashed,ed.),Academic Press.

—— (ed.)(1976):'Generalized inverses and applications',(Academic Press).

—— (ed.)(1979):'Functional analysis methods in numerical analysis', (Springer LNM,No.701).

NEHARI,Z.(1952):In,Beckenbach,(ed.),'Construction and applications of conformal maps',(N.B.S.Series in Applied Mathematics,1952).

NORRIE,D. and G.de VRIES(1976):'Finite element bibliography',(IFI/Plenum,New York).

NOLAN,J.(1953):'Analytical differentiation on a digital computer', M.A.Diss.M.I.T..

NEWELL,J.C.et al.(1957):'Empirical explorations of the logic theory machine: a case study in heuristics',Report P-951,RAND Corporation, 48 pages.

NEWMAN,C.M.(1974):'Zeros of the partition function for generalized Ising systems', Commun.Math.Phys.,Volume 27,143-159.

NG(1979):'Symbolic/Numeric Interface',in,Proc.'EUROSAM '79'(Springer LNCS 72), pages 330-345.

NEY,P.(1970):'Advances in probability,Volume 1',(Marcel Dekker).

—— and S.PORT(1974): ——— Volume 2,(Marcel Dekker).

NICKEL,K.(1975):'Interval mathematics',(Springer LNCS,No.29).

ODEN,J.T.(1972):'Finite elements for nonlinear elastic continua',(McGraw-Hill).

OLSEN,J.et al.(1978-):(Computer programs for determinacy and unfolding calculations), Dept.Comp.Sci.,Brigham Young Univ.;repr.as an appendix to Poston and Stewart,'Catastrophe Theory',(Pitman,1978).

O'MALLEY,R.(1974):'Introduction to singular perturbations',(Academic Press).

ORTEGA, .and .RHEINBOLDT(1970):'Iterative solution of nonlinear equations in several variables',(Academic Press).

OSTROWSKI,A.(1946):'Sur les relations algébriques entre les intégrales indéfinies', Acta Math.,Vol.78,315-318.

—— (1952): Two papers in,Beckenbach,E.,ed.,'Construction and applications of conformal maps',N.B.S.Appl.Math.Series,1952.

—— (197):'Solution of equations and systems of equations in Banach spaces',(Academic Press).

PATIL,D.J.(1972):'Representation of H^p-functions',Bull.AMS,18(4),617-620.

PATTERSON,W.M.(1974):'Iterative methods for the solution of a linear operator equation in a Hilbert space—a survey',(Springer LNM,No.394).

PENROSE,O.(1970):'Foundations of statistical mechanics',(Pergamon Press).

—— and ELVEY,J.S.N.(1968):'The Yang-Lee distribution for a classical one-dimensional system',J.Phys.A,Series 2,661-674.

PEYERIMHOFF,A.(1969):'Lectures on summability',(Springer,LNM,No.107).

PETROV,V.V.(1973):'Sums of independent random variables',(Springer).

PERISHO,R.C.(1975):'ASHMEDAI User's Guide',Report No.COO-3066-44,U.S.A.E.C..

PETERSEN,G.M.(1966):'Regular matrix transformations',(McGraw-Hill).

PHAN,D.D.,(1974):'Some questions in constructive functional analysis', Proc.Steklov Inst.Math.,No.114

PHELPS,R.R.(1966):'Lectures on Choquet's theorem',(van Nostrand).

PICONE,M.(1954):'Sul calcolo delle funzioni olomorphe di una variabile complessa', in,'Studies in Math.and Mech.Pres.to R.von Mises'(Academic Press).

PITT,H.(1958):'Tauberian theorems',(Oxford University Press).

PIMBLEY,G.(1969):'Eigenfunction branches of nonlinear operators and their bifurcations',(Springer LNM,No.104).

POSTON,T.and I.N.STEWART(1976):'Taylor expansions and catastrophes', (Pitman Research Notes in Math.,No.7).

—— and —— (1978):'Catastrophe Theory and its Applications', (Pitman:Surveys and Ref.Works in Math.Series).

PREISENDORFER,R.(1965):'Radiative transfer in discrete spaces',(Pergamon Press).

PRODI,G.(1971):'Problems in nonlinear analysis',(C.I.M.E.,IV Ciclo,Varenna,(8/70)).

RABINOWITZ,P.,(ed.)(1977):'Applications of bifurcation theory',(Academic Press).

RALL,L.(1969):'Computational solution of nonlinear operator equations', (Prentice-Hall).

REISSIG,R.,SANSONE,G.and R.CONTI(1974):'Nonlinear differential equations of higher order',(Noordhoff).

REICHENBACH,H.(1949):'Theory of probability',2nd Ed.,(U.of California Press).

RÉMOUNDOS,G.(1927):'Extensions aux fonctions algébroïdes multiformes du théorème de M.Picard',(Gauthier-Villars).

RENYI,A.and R.SULANKE(1963):'Über die konvexe Hulle von n zufällig gewälten Punkten,I.',Z.Wahrschein.Vol.2,75-84;also,II.,Z.Wahrschein.,Vol.3,138-147.

RICE,J.(1969):'The approximation of functions.Volume 2.Nonlinear and multivariate theory',(Addison-Wesley).

RICH,A.D. and D.R.STOUTEMYER(1979):Proc.'EUROSAM '79',(Springer LNCS,72),241-248.

RESCHER,N.(1969):'Many-valued logic',(McGraw-Hill).

RICHARDSON,D.(1968):'Some undecidable problems involving Elementary functions of a real variable',J.Symb.Logic,Vol.33,514-520.

——— (1971):'A solution of the identity problem for integral exponential functions',Z.Math.Logik u.Grundl.Math..

RISCH,R.H.(1969):'The problem of integration in finite terms',Trans.AMS,139,167-189.

——— (1970):'Solution of the problem of integration in finite terms', Bull.AMS,May,1970,605-608.

——— (1976):'Implicitly elementary integrals',Proc.AMS,57,1-7.

——— (1979):'Algebraic properties of the Elem.Fns. ...',Amer.J.Math.,101,p743

RITT,J.F.(1948):'Integration in finite terms',(Columbia University Press).

——— (1950):'Differential algebra',(AMS Colloq.Publ.;repr.,Dover,1966).

ROBINSON,D.W.(19):

ROSENBLAT,S.(1979):Studies in Appl.Math.

ROTHSTEIN,M.(1976):'Aspects of symbolic integration and simplification of exponential and primitive functions',Ph.D.Diss.,Univ.Wisconsin, Madison.

RUELLE,D.(1969):'Statistical mechanics:rigorous results',(Benjamin,New York).

——— (1980):In,Mathematical Intelligencer,Vol. (ed.H.M.Edwards,NY).

——— (1973):'Some remarks on the location of zeros of the partition function for lattice systems',Commun.Math.Phys.,31,265-277.

RUNNELS,L.K.and J.B.HUBBARD(1971):'Applications of the Yang-Lee-Ruelle theory to hard-core lattice gases',J.Statist.Phys.Vol.6(1),1-20.

RUSSELL,B. and A.N.WHITEHEAD(1910):'Principia Mathematica',(Cambridge Univ.Press).

SAATY,T.L.(1967):'Modern nonlinear equations',(McGraw-Hill).

SAMMET,J.E.(1967):Adv.Comp.,Vol.8,47-102.

---- (1969):'Programming languages:history and fundamentals',(Prentice-Hall).

---- (1971):In,'Mathematical Software',(Academic Press).

SAKS,S. and A.ZYGMUND(1971):'Analytic functions',(PWN Publ.,Warsaw).

SANIN,N.A.(1964):In,AMS Translations,Series 2,Vol.99,233-275(1972).

SARD,A.(1942):'The measure of critical values of differentiable maps', Bulletin AMS,Vol.48,883-890.

---- (1963):'Linear approximation',AMS Math.Surveys,No.9.

SCHMIDT,E.(1908):Math.Ann.,Vol.65.

SCHMIDT,P.(1976):'Automatic symbolic solution of differential equations of first order and first degree',Proc.1976 ACM Symp.Symb.Alg.Comp., Yorktown Heights,NY,114-125.

------ (1979):'Substitution methods for the automatic symbolic solution of differential equations of first order and first degree', Proc.'EUROSAM '79',(Springer LNCS,72),164-176.

SCHAEFER,M.(1973):'A mathematical theory of global program optimization', (Prentice-Hall).

SCHOENBERG,I.J.(1968):'Approximations,with emphasis on spline functions', (Academic Press).

SCHWARTZ,L.(1950-1951):'Théorie des Distributions',(Hermann).

SCHWEIZER,B.and .SKLAR(1960):'Statistical metric spaces',Pacif.J.Math.,10,313-334.

SCOTT,D.(1976):'Is many-valued logic any use?',from,'Philosophy of logic', S.Körner,ed.,(Basil Blackwell,Oxford).

SELBERG,H.(1934):Avhandl.Norske Videnskaps-Acad.,Oslo,Vol.8,1-72.

SHAMOS,M.I.(1977):'Geometry and statistics:problems at the interface', in,'Algorithms and complexity',J.F.Traub,ed.,Acad.Press,251-280.

------ (1978):'Computational geometry',Ph.D.Thesis,Yale University;to appear as,'Computational geometry',Springer-Verlag.

SHERMAN,D.I.(1940):Dokl.Vol.XXVII(9),911-913;XXVIII(1),25-28.

SHOWALTER,W.(1977):'Hilbert space methods for partial differential equations', (Pitman).

SHTOKHAMER,R.(1975):'Simplification of nested radicals',Univ.of UtahComp. Phys.Group,Report UCP-37

SINGER,I.(1970a):'Bases in Banach spaces',(Springer,Grundlehren).

---- (1970b):'Best approximation of elements in Banach spaces... ',(Springer).

SIROVICH,L.'Techniques of asymptotic analysis',(Springer,Lec.Notes Appl.Math.).

SMART, .(19):'Fixed-point theorems',(Cambridge Tract).

SMIRNOV,V.and N.A.LEBEDEV(1968):'Functions of a complex variable:constructive theory',(Iliffe).

SOBOLEV,S.L.(1936):Mat.Sbornik (N.S.),Vol.1,39-71.

SKOLNIKOFF,I.(1956):'Mathematical theory of elasticity',(McGraw-Hill).

SOUTHWELL,R.V.(1946):'Relaxation methods in engineering science,Vol.1',Oxford U.P..

SPRINGER,G.(1957):'Introduction to Riemann surfaces',(Addison-Wesley).

STEIN,P.R.and S.M.ULAM(1964):'Nonlinear tranformation studies on electronic computers',(Warsaw,1964);repr.in,'Stanislaw Ulam:Sets Numbers and Universes',(M.I.T.Press,1974).

STOUTEMYER,D.(1974a):'Analytical optimization using algebraic manipulation', U.Hawaii ,The Aloha System,TR A74-3.

------ (1974b):'Computer algebraic manipulation for the calculus of variations,the maximum principle and automatic control', U.Hawaii,The Aloha System,TR A74-5.

------ (1975):'Analytical optimization using computer algebraic manipulation', ACM Trans.Math.Software,Vol.1,147-164.

------ (1977a):'Analytically solving integral equations by using computer algebra',ACM Trans.Math.Software,Vol.3,128-146.

------ (1977b):'Automatic error analysis using computer algebraic manipulation',ACM Trans.Math.Software,Vol.3,26-43.

------ (1977c):'Symbolic computer vector analysis',Proc.1977 MACSYMA Users' Conference,(NASA Report CP-2012),447-460.

SUNDBLAD,Y.(1974):'Symbolic manipulation systems,now and in the future', Proc.'EUROSAM '74',SIGSAM Bulletin No.31,1-8.

STRANG,G.and G.FIX(1973):'An analysis of the finite element method',(Prentice-Hall).

STUMMEL,F.(1973):'Approximation methods in analysis',Lec.Notes,Univ.Aarhus.

------ (1975):'Discretely uniform approximation of continuous functions', J.Approx.Theory,Vol.13,178-191.

------ (1979):'Nonconforming finite element methods and related approximation methods.Hauptvortrag,'Meth.Nichtkonformen.fin.el.',GAMM-Tagung,Wiesbaden.

STRUBBE,H.(1974):'Manual for SCHOONSHIP:a CDC 6000/7000 program for symbolic evaluation of algebraic expressions',Comp.Phys.Comm.Vol.8,1-30.

SUZUKI,M. and M.E.FISHER(1971):'Zeros of the partition function for the Heisenberg, Ferroelectric and general Ising models',J.Math.Phys.Vol.12(2).

SYMM,G.(1966):'An integral equation method in conformal mapping',Num.Math., Vol.9,250-258. (Also:Num.Math.,Vol.10,437-445(1967)).

---- (1969):'Conformal mapping of doubly-connected domains',Num.Math., Vol.12,448-457.

TIKHONOV,A.N. and .ARSENIN(1977):'Solution of ill-posed problems',(Winston/Wiley).

TIMOSHENKO,S. and .GOODIER(1970):'Theory of elasticity',3rd Ed.(McGraw-Hill).

TITCHMARSH,E.C.(1948):'Intro.to the theory of Fourier integrals',2nd Ed.,Oxford U.P.

TRAGER,B.(1979):'Integration of simple radical extensions',Proc.'EUROSAM '79', (Springer,LNCS,72),408-414.

TRAUB,J.F.,(ed.)(1976):'Analytic computational complexity',(Academic Press).

---- (ed.)(1977):'Algorithms and complexity',(Academic Press).

---- and H. WOZNIAKOWSKI(1980):'Analytic computational complexity',(Academic Press)

TREVES,F.(1967):'Linear partial differential equations with constant coefficients', (Gordon and Breach).

TOBEY,R.G.(1971):'Symbolic mathematical computation—introduction and overview', Proc.Second Symp.Symb.alg.manip.(ACM,1971),1-16.

TODD,M.J.(1976):'The computation of fixed points and applications' Springer Lec.Notes in Econ.and Math.Systems,No.124.

TROTMAN,D. and E.C.ZEEMAN(1973):'The classification of elementary catastrophes of codimension $\leqslant 5$ '.Revised form repr.in 'Catastrophe theory:selected papers',by E.C.Zeeman,(Addison-Wesley,1977),497-561.

TSUJI,M.(1959):'Potential theory in modern function theory',(Maruzen;repr.,Chelsea).

UHLENBECK,G.E. and G.FORD(1962):In,Part B,of 'Studies in statistical mechanics,Vol,1', J.de Boer and G.E.Uhlenbeck,eds.,(North-Holland).

VAINBERG,M.M. and V.A.TRENOGIN(1974):'Theory of branching of solutions of nonlinear equations',(Noordhoff).

VISCONTI,A.(1969):'Quantumfield theory,Vol.1',(Pergamon Press).

VOROS,A.(1977):Doctoral thesis,(Univ.Paris-Sud,Orsay).

—— (1979):'Symbol calculus by symbolic computation and semi-classical expansions',Proc.'EUROSAM '79',(Springer LNCS,72),45-51.

WAERDEN,B.L.van de(1949):'Modern algebra',(Ungar,New York).

WALSH,J.L.(1950):'The location of critical points',AMS Colloq.Publ.,No.34.

—— (1966):'Interpolation and approximation by rational functions in the complex domain',4th Ed.of AMS Colloq.Publ.No.20.

WACHPRESS,E.(1975):'A rational basis for finite elements',(Academic Press).

WANG,H.(1960):'Toward mechanical mathematics',IBM J.Res.and Dev.,Vol.4.,(also,in WANG,H.(1963)).

WANG,H.(1963):'A survey of mathematical logic',(Science Press,Peking/North-Holland).

WANG,P.(1971):'Evaluation of definite integrals by symbolic manipulation', Report 92,Project MAC,M.I.T..

—— (1976):'Factoring multivariate polynomials over algebraic number fields', Math.of Comp.,Vol.30,No.134,324-336.

—— and ROTHSCHILD,L.(1975):'Factoring multivariate polynomials over the integers',Math.of Comp.,Vol.29,935-950.

WEISSMAN,C.(1967):'LISP 1.5 Primer',(Dickinson Publ.Co.,Belmont,California).

WEIL,A.(19):

WEISEL,J.(1979):Mitteilungen aus dem Mathem.Seminar Giessen,Heft 138.

WENDLAND,W.(1979):'Linear elliptic systems in the plane',(Pitman).

WHITE,J.(1977):'LISP:Program is data.A historical perspective on MACLISP', Proc.1977 MACSYMA Users' Conf.(NASA Report CP-2012),181-189.

WIENER,N.(1923):'Differential space',J.Math.and Phys.,Vol.2,131-174.

—— (1932):'Tauberian theorems',Ann.Math., (2),33,1-100.

WHITEMAN,J.,(ed.)(1975):'Mathematics of finite elements and applications,Vol.1', (Academic Press).

—— (ed.)(1977): ———— Vol.2.,(Academic Press).

—— (ed.)(1979): ———— Vol.3.,(Academic Press).

WILSON,K.(1973):Cargese Lecture Notes(theoretical physics).

WILLSON,A.,(ed.)(1974):'Nonlinear networks:theory and analysis',(IEEE Press).

WONG,E. and M.ZAKAI(1965a):'On the relation between ordinary and stochastic differential equations',Internat.J.Engrg.Sci.,3,213-229.

—— (1965b):'On the convergence of ordinary integrals to stochastic integrals',Ann.Math.Statistics,36,1560-1564.

—— (1965c):'On convergence of the solutions of differential equations involving Brownian motion', Report No.65-5,AF-AFOSR-139-64,Electronics Res.Lab., Univ.of California,Berkeley.

YANG,C.N. and T.D.LEE(1952):'Statistical theory of equations of state and phase transitions,I.',Phys.Rev.,87,404.

YOUNG,L.C.(1970):'Some new stochastic integrals and Stieltjes integrals,Part I', in,Adv.in Probability,P.Ney,ed.,(Dekker,1970),161-240.

—— (1974): ——— Part II,in,Adv.in Prob.,P.Ney and S.Port,eds., (Dekker,1974),101-178.

ZADEH,L.(1965):'Fuzzy sets',Information and Control,Vol.8,338-353.

ZASSENHAUS,H.(1969):'On Hensel factorization',J.Number theory,Vol.1,291-311.

ZEEMAN,E.C.(1964):'On relative simplicial approximation',Proc.Cam.Phil Soc.60,39-43.

—— (1977):'Catastrophe theory:selected papers,1972-1977',(Addison-Wesley).

ZIN,G.(1953):Annali Mat.Pura Appl.,(4),34,365-405.

ZIPPEL,R.(1977):'Radical simplification made easy',Proc.1977 MACSYMA Users' Conf., (NASA Report CP-2012),361-367.

ZUBOV,V.I.(1962):'Mathematical methods for the study of automatic control systems',(Pergamon Press/Jerusalem Academic Press).

[BAR 68] Bareiss, E.H., Math. of Comp., Vol. 22, No. 103 (July 1968), pp. 565-578.

[BER67] Berlekamp, E. R., Factoring Polynomials over Finite Fields, Bell System Tech. J., Vol. 46 (1967), pp. 1853-1859.

[BER68] Berlekamp, E. R., Algebraic Coding Theory, McGraw-Hill, 1968

[BRO63] Brown, W. S., The ALPAK System for Non-numerical Algebra on a Digital Computer - 1: Polynomials in Several Variables and Truncated Power Series with Polynomial Coefficients, Bell System Tech. J., Vol. 42, No. 3 (Sept. 1963), pp. 2081-2120.

[BRO68] Brown, W. S., The Compleat Euclidean Algorithm, Bell Telephone Laboratories Report, June 1968, 68 pages.

[COL60] Collins, George E., A Method for the Overlapping and Erasure of Lists, Comm. A.C.M., Vol. 3, No. 12 (Dec. 1960), pp. 655-657.

[COL66] Collins, George E., PM, A System for Polynomial Manipulation, Comm. A.C.M., Vol. 9, No. 8 (Aug. 1966), pp. 578-589.

[COL67] Collins, George E., Subresultants and Reduced Polynomial Remainder Sequences, Jour. A.C.M., Vol. 14, No. 1 (Jan. 1967), pp. 128-142.

[COL67b] Collins, George E., The SAC-1 List Processing System, University of Wisconsin Computing Center Technical Report, July 1967, 34 pages.

[COL68a] Collins, George E., and James R. Pinkert, The Revised SAC-1 Integer Arithmetic System, University of Wisconsin Computing Center Technical Report No. 9, Nov. 1968, 50 pages.

[COL68b] Collins, George E., The SAC-1 Polynomial System, University of Wisconsin Computing Center Technical Report No. 2, March 1968, 68 pages.

[COL68c] Collins, George E., The SAC-1 Rational Function System, University of Wisconsin Computing Center Technical Report No. 8, July 1968, 31 pages.

[COL69] Collins, George E., Computing Multiplicative Inverses in GF(p), Math. Comp., Vol. 23, No. 105 (Jan. 1969), pp. 197-200.

[COL69b] Collins, George E., Algorithmic Approaches to Symbolic Integration and Simplification (Summary of the 1968 FJCC Panel Session), SIGSAM Bulletin, No. 12 (July 1968), pp. 5-16.

[COL69c] Collins, George E., L. E. Heindel, E. Horowitz, M. T. McClellan, and D. R. Musser, the SAC-1 Modular Arithmetic System, University of Wisconsin Technical Report No. 10, June 1969, 50 pages.

[COL69d] Collins, George E., Computing Time Analyses for Some Arithmetic and Algebraic Algorithms, Proceedings of the 1968 Summer Institute on Symbolic Mathematical Computation (Robert G. Tobey, ed.), IBM Federal Systems Center, June 1969, pp. 195-232.

[COL70] Collins, George E., and Ellis Horowitz, The SAC-1 Partial Fraction Decomposition and Rational Function Integration System, University of Wisconsin Computing Center Technical Report No. 12, Feb. 1970, 47 pages.

[COL70b] Collins, George E., and Lee E. Heindel, The SAC-1 Polynomial Real Zero System, University of Wisconsin Computing Center Technical Report No. 18, Aug. 1970.

[COL70c] Collins, George E., The Calculation of Multivariate Polynomial Resultants, Proc. of the Second Symposium on Symbolic and Algebraic Manipulation, Los Angeles, March 1971.

[COL70d] Collins, George E., The SAC-1 Polynomial Greatest Common Divisor and Resultant System. In preparation.

[COL70e] Collins, George E., and David R. Musser, The SAC-1 Polynomial Factorization System. In preparation.

[COL70f] Collins, George E., and Michael T. McClellan, The SAC-1 Polynomial Linear Algebra System. In preparation.

[HAR16] Hardy, G. H., The Integration of Functions of a Single Variable, second ed., Cambridge Univ. Press, 1916.

[HEI70] Heindel, Lee E., Algorithms for Exact Polynomial Root Calculation, Ph.D. Thesis, University of Wisconsin Computer Sciences Department, July 1970, 153 pages.

[HEN56] Henrici, P., A Subroutine for Computations with Rational Numbers, Jour. A.C.M., Vol. 3, No. 1 (Jan. 1956), pp. 6-9.

[HOR69] Horowitz, Ellis, Algorithms for Symbolic Integration of Rational Functions, Ph.D. Thesis, University of Wisconsin Computer Sciences Department, Nov. 1969, 132 pages.

[HOR70] Horowitz, Ellis, Algorithms for Partial Fraction Decomposistion and Rational Function Integration, University of Wisconsin

Some basic references for SAC-1.(From Collins(1971)).

Computer Sciences Department Technical Report No. 91, June 1970, 44 pages.

[JOH66] Johnson, S. C., Tricks for Improving Kronecker's Polynomial Factoring Algorithm, Bell Telephone Laboratories Report, 1966.

[JOR66] Jordan, D. E., R. Y. Cain, and L. C. Clapp, Symbolic Factoring of Polynomials in Several Variables, Comm. A.C.M., Vol. 9, No. 8 (Aug. 1966), pp. 638-643.

[KNU68] Knuth, D. E., The Art of Computer Programming, Vol. 1 (Fundamental Algorithms), Addison-Wesley, 1968.

[KNU69] Knuth, D. E., The Art of Computer Programming, Vol. 2 (Seminumerical Algorithms), Addison-Wesley, 1969.

[LIP69] Lipson, John D., Symbolic Methods for the Computer Solution of Linear Equations with Applications to Flowgraphs, Proceedings of the 1968 Summer Institute on Symbolic Mathematical Computation (Robert G. Tobey, ed.), June 1969, IBM Federal Systems Center, pp. 233-303.

[MAN68] Manove, M., S. Bloom, and C. Engelman, Rational Functions in MATHLAB. In Bobrow, D. G. (Ed.), Symbol Manipulation Languages and Techniques, North Holland, Amsterdam, 1968, pp. 86-102.

[MCC71] McClellan, Michael T., Algorithms for the Exact Solution of Systems of Linear Equations with Polynomial Coefficients, University of Wisconsin Computer Sciences Department Ph.D. Thesis. In preparation.

[MUS71] Musser, David R., Algorithms for Polynomial Factorization, University of Wisconsin Computer Sciences Department Ph.D. Thesis. In preparation.

[RIS69] Risch, Robert H., Symbolic Integration of Elementary Functions, Proceedings of the 1968 Summer Institute on Symbolic Mathematical Computation (Robert G. Tobey, ed.), June 1969, IBM Federal Systems Center, pp. 133-147.

[TAR51] Tarski, A., A Decision Method for Elementary Algebra and Geometry, University of California Press, 1951 (Second ed., rev.).

[TOB67] Tobey, Robert G., Algorithms for Anti-Differentiation of Rational Functions, Ph.D. Thesis, Harvard University, 1967.

[VDW43] Van der Waerden, B. L., Modern Algebra, Vol. I, Frederick Ungar Publishing Co., 1948.

[ASA 64] American Standards Association, A Programming Language for Information Processing on Automatic Data-Processing Systems
Comm. ACM, Vol.7, No.10 (Oct., 1964) pp591-625.

NOTE THAT, although these references only go to 1970, the basic hierarchical structure of SAC-1 (as reflected in the subsystems) is essentially covered. This situation contrasts with that of FORMAC, where the early versions do not give an adequate idea of the present strength of the system.
Current information on SAC-1 may be obtained from the Technical Reports (in their latest forms) and general enquiries may be sent to G.E. Collins at the University of Wisconsin, Madison.