Analytic calculations on digital computers for applications in physics and mathematics

V. P. Gerdt, O. V. Tarasov, and D. V. Shirkov

Joint Institute for Nuclear Research, Dubna, Moscow District
Usp. Fiz. Nauk **130**, 113–147 (January 1980)

The present state of analytic calculations on computers is reviewed. Several programming systems which are used for analytic calculations are discussed: SCHOONSCHIP, CLAM, REDUCE-2, SYMBAL, CAMAL, AVTO-ANALITIK, MACSYMA, etc. It is shown that these systems can be used to solve a wide range of problems in physics and mathematics. Some physical applications are discussed in celestial mechanics, the general theory of relativity, quantum field theory, plasma physics, hydrodynamics, atomic and molecular physics, and quantum chemistry. Some mathematical applications which are discussed are evaluating indefinite integrals, solving differential equations, and analyzing mathematical expressions. This review is addressed to physicists and mathematicians working in a wide range of fields.

PACS numbers: 89.80. + h, 02.70. + d

CONTENTS

1. INTRODUCTION

The idea that any exact analytic procedure could be performed by a computer was expressed back in 1844 by Lady Lovelace (Ada Augusta), the patron of the English mathematician Charles Babbage. In 1833 Babbage developed an "analytical engine,"[1] which was a programmable calculating machine with arithmetic and memory devices. A hundred years later, these properties of Babbage's machine became the foundation of modern computers. The first successful attempts to perform analytic calculations on a computer were undertaken a quarter-century ago for the simplest nonnumerical operation: differentiation.[2] The next important step in this direction—the development of a polynomial algebra—required another ten years of rapid developments in computer technology, the development of new nonumerical algorithms,[3] and the creation of algorithmic programming languages. Among these languages are the familiar[4] FORTRAN, ALGOL, PL/1, etc.; although LISP,[5] devised by McCarthy[6] in 1960, is less familiar, it is far better suited for analytic calculations.

Since the mid-1960's, more than 30 programming systems have been devised for analytic calculations; of these, about ten have found extensive use on a wide range of problems in physics and mathematics.

Nevertheless, many people are extremely surprised to hear that computers are capable of carrying out analytic as well as numerical calculations. The surprise stems from both the general view of computers as tools designed especially for numerical calculations and the extreme paucity of information on analytic computer calculations in the Soviet literature.

In algorithmic languages for numerical programming, such as FORTRAN and ALGOL, all the symbols which are used of course represent certain numbers, and the result of the calculation is also a number.

With the programming systems for analytic calculations, on the other hand, it becomes possible to use computers to perform analytically such operations as differentiation, simplification of expressions (collection of similar terms), replacement of a symbol or expression by another expression, etc. The net result of the calculation—and this point deserves particular emphasis—is some analytic expression, e.g., a function with an explicit dependence on its arguments.

It is remarkable that analytic calculations can be carried out on digital computers, and that this capability was attained only after the development of several subtle programming methods. In numerical calculations, for example, the particular way in which the data are distributed in the computer memory is usually

 0038-5670/80/010059-19$01.10

fixed, and the amount of data is known before the program is begun. For analytic calculations, in contrast, it is difficult, and sometimes impossible, to allot the memory beforehand or to determine accurately how much memory will be required for the intermediate steps. One way to approach this problem is to use a dynamic allocation of the memory: In the intermediate steps of the calculations, data which are no longer required are erased, and the memory which thus becomes available is used to store new data. The algorithmic language LISP is an excellent example of an implementation of this principle.

Another distinctive feature of analytic calculations is the greater complexity of the elementary operations (adding, multiplying, etc.), which must be modified by programming. Furthermore, the instructions required to control both the input data and the calculated results are more complicated. This is the primary distinction between analytic and numerical calculations.

In some cases, analytic calculations can be carried out without making direct use of symbols, which, like numbers, are easily stored in the computer memory in binary code. For example, the polynomial $(54/7)x^3yz^2$, expressed in terms of the variables x,y,z,w, can be represented completely unambiguously by the set of numbers 54, 7, 3, 1, 2, 0. This representation makes it a simple matter to develop a compact and rapid system, but when we turn to a symbolic operation, even such a simple one as differentiation, the programming becomes extremely difficult.

At present we may distinguish among four methods for carrying out analytic calculations on computers; these methods of course have several features in common.

First method. The input data are entered into the computer memory in some compact way, like that described above, and the program required for controlling these data is written in a low-level language: an assembler language (which is a symbolic form of the machine instructions). The first method is usually used to solve a narrow range of problems, and it is the basis for many specialized analytic programming systems which are compact and fast. One example is the SCHOONSCHIP[7] system. This first approach suffers from the drawback, however, that much tedious labor is required of highly skilled programmers in order to introduce any new capabilities (i.e., any new mathematical operations).

Second method. Again, the input data are stored in a compact manner, but the program for manipulating these data either is written entirely in a higher-level language (e.g., FORTRAN) or takes the form of subprograms, some in the higher-level language and some in an assembler language. These subprograms are controlled in the higher-level language. One system which uses this method is the SAC-1 system.[8]

Third method. In this case the necessary operations are programmed directly in a language at a higher level than the assembler language. In this case, both the input data and the manipulations of these data are governed by the syntax of the particular language used. In this method, the internal representation of the input data is no longer necessarily compact, as it is in the first two methods. Another distinguishing feature of this method is that there is no tie with the assembler, which differs from computer to computer. It is thus relatively painless to transfer a program written in this manner from one type of computer to another. The language used for this purpose may be, for example, LISP. In LISP, for example, it is a simple matter to write a symbolic-differentiation program,[9] using a table of derivatives.

Fourth method. This is the most promising method for analytic calculations, and it is the basis for the most sophisticated analytic programming systems. This method involves the use of a set of standard analytic-manipulation subprograms, written in any symbolic language L (for example, LISP). Furthermore, the system incorporate a large number of special functions, which are written out beforehand in L. The input data and the program for controlling them are written in a special external language, developed along with the rest of the system. This external language must be of maximum convenience for the user, among other things. Several of the systems which have been developed thus have, for example, an ALGOL-like external language (Table I). An important advantage of this latter approach is that the user himself can extend the capabilities of the system by adding the subprograms which he needs, written in the external language or L. Although convenient for writing programs, these systems are generally slower than systems of other types.

This fourth method for analytic programming is used in some universal analytic programming systems, in particular, REDUCE-2 (Ref. 10). The process for ex-

TABLE I. General properties of certain programming systems for analytic calculations.

System →	SCHOONSCHIP	CLAM	REDUCE-2
Version (year)	1977	1972	1973
Computer	CDC-6500	CDC-6500	CDC-6500 EC-1040
Capacity of system	25000 words	20000 words	65000 words (CDC) 300 kbytes (EC)
Status at JINR	Adopted	Adopted	Adopted
Implementation language	Assembler	Assembler	LISP
External language	*	LISP	ALGOL
Interactive mode	Yes	No	Yes
Primary applications	QFT	GTR	Universal

System →	SYMBAL	CAMAL	AVTO-ANALITIK	MACSYMA
Version (year)	1970	1975	1973	1977
Computer	CDC-6500	EC-1040	BÉSM-6	DEC PDP-10
Capacity of system	25000 words	240 kbytes	30000 words	221000 words
Status at JINR	Adopted	Being adopted	BÉSM-6	–
Implementation language	Assembler	BCPL	Machine Code	LISP
External language	ALGOL	*	*	ALGOL
Interactive mode	No	No	No	Yes
Primary applications	GP	CM and GTR	MP	Universal

*Special language of this system; QFT) quantum field theory; GTR) general theory of relativity; CM) celestial mechanics; MP) mathematical physics; GP) general purpose.

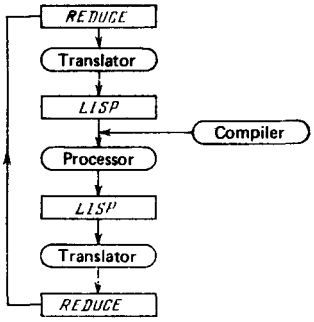

FIG. 1. Diagram for executing the instructions in the REDUCE-2 system.

ecuting an instruction in the REDUCE-2 system is illustrated by the diagram[11] in Fig. 1. It is clear that this multiple-step translation and the use of the LISP system affect the speed of REDUCE-2.

A computer can be of assistance, of course, only if the procedure for finding a solution is clear, i.e., only if a clearly defined algorithm is available for constructing the necessary solution. An analytic programming system can be thought of as a powerful and essentially unique tool for solving the following two types of problems: 1) those problems which require an unacceptable amount of manual calculation; 2) those which are very sensitive to a loss of accuracy in numerical solutions.

Among the problems of the first category is that of inverting a high-order matrix whose elements are symbols or algebraic expressions.

An important problem in the second category is that of analyzing the plasma stability in a tokamak; this problem reduces to one of determing the condition under which some function is zero in a specified region. The position of this zero is very sensitive to a loss of accuracy in intermediate calculations. Analytic programming systems have proved exceptionally useful in this problem, and this case will be discussed in detail below (Chapter 3).

2. GENERAL PROPERTIES OF PROGRAMMING SYSTEMS FOR ANALYTIC CALCULATIONS

For our purposes we can classify all the existing analytic programming systems in three groups.

I. Specialized systems. These systems have been developed for a definite field of application and usually for very long calculations. For many systems of this type, the number of mathematical operations involved is small. These operations generally use special algorithms, so that the calculations can be carried out rapidly, with a relatively small demand on the central computer memory.

Among these specialized analytic programming systems are SCHOONSCHIP[7] and ASHMEDAI,[12] designed for standard calculations in quantum field theory; ALAM,[13] its modification CLAM,[14] and also SHEEP,[15] which have been developed for calculations in the general theory of relativity; MAO[16] and AMS,[17] which have been developed for operations with Poisson series in stellar mechanics; CAMAL,[18] developed for problems of celestial mechanics and the general theory of relativity; and AVTOANALITIK,[19] developed for solving several problems of mathematical physics.

II. General-purpose systems. These incorporate the mathematical operations typical of many problems in physics and mathematics; differentiation, vector algebra, matrix algebra, etc.

Among the systems we will mention FORMAC,[20] ALTRAN,[21] SYMBAL,[22] and ANALITIK-74 (Ref. 23).

III. Universal systems. Three of the most sophisticated general-purpose systems fall in this group: REDUCE-2 (Ref. 10), MACSYMA,[24], and SCRATCHPAD.[25] The last two of these are the most powerful of the programming systems for analytic calculations now in existence. They can handle the overwhelming majority of analytic operations which can be carried out on present-day computers. The REDUCE-2 system incorporates many fewer operations than does MACSYMA or SCRATCHPAD. However, the user of REDUCE-2 has much flexibility in defining new objects and/or new mathematical operations in the external language of the system. This particularly attractive feature of REDUCE-2 is the stimulus for its continuous development and puts it in the class of universal systems. We should emphasize here that many specialized systems and some of the general-purpose ones are written in an assembler language, so that they tend to be compact and fast. The universal systems, on the other hand, are written in LISP.[5] Although LISP is excellent for analytic calculations, it requires a comparatively large amount of computer time to carry out the individual operations. As a result, the calculations by the universal systems are relatively "slow."

In contrast with numerical programs, analytic systems, especially the universal ones, require a large computer memory—typically tens of thousands and sometimes hundreds of thousands of machine words (see, for example, Table I). In the actual use of any system, of course, more memory is required, the precise amount depending on the type of problem to be solved. This severe demand on the computer memory is a consequence of the particular nature of the nonnumerical algorithms and the need to store all the intermediate results in the memory. These intermediate results frequently expand in volume greatly in the course of a calculation. The most powerful analytic programming systems are accordingly used on the larger computers, e.g., the IBM 360/370, the CDC 6000/7000, and the DEC PDP-10. An excellent example of an analytic programming system for small computers is ANALITIK-74 (Ref. 23), developed in the Soviet Union for MIR-3 computers. Unfortunately, despite the flexibility of ANALITIK and its use for a variety of problems,[26] it is limited to comparatively modest calculations on the MIR-3. In the Soviet Union, research on "computer analytics" was begun in the late 1950's by Kantorovich and his students.[27-31] (see also the book by Smirnova[32] and the bibliography there) and later by Shurygin and Yanenko.[33] Several analytic programming

systems have now been developed in the Soviet Union for Soviet computers: the SIRIUS system[34] for the M-20 and M-222 computers; the AVTO-ANALITIK and AMS, mentioned above, for the BESM-6 computer, the ANALITIK-74 system, etc. In addition to systems, several processors and special programs have been developed, for example, the KINO and PASSIV processors for analyzing the group properties of differential equations[35] and programs which perform certain calculations in quantum electrodynamics.[36,37] A good picture of the various Soviet-trends in computer analytics is drawn by the proceedings of the All-Union Discussion of the Use of Digital Computers for Analytic Calculations.[38]

The use of analytic programming systems in the Joint Institute for Nuclear Research began in 1975 with the asoption of the SCHOONSCHIP system.[7] Since then, the CLAM,[14] REDUCE-2 (Ref. 10), and SYMBAL[22] systems have been asopted; CAMAL[18] is in the process of being adopted; and plans call for the adoption of the AVTO-ANALITIK system[19] in the near future. Tables I and II list the most general properties of these systems, along with the mathematical objects and the operations on these objects (these are also typical of many other analytic programming systems). Shown for comparison in the last column of each table is the information corresponding to the *MACSYMA* system[24]—which is the most powerful of the existing analytic programming systems. Among the general properties (Table I) we wish to single out the interactive mode of operation, which is embodied in the SCHOONSCHIP and REDUCE-2 systems. When working in this mode, the user can monitor the intermediate results and thus greatly streamline the solution of many problems. Among the capabilities incorporated in analytic programming systems (Table II) we have not mentioned *polynomial algebra*, *simplification* of expressions (the collection of similar terms), and *substitution*, which are characteristics of all modern systems. In certain cases, the substitution capability makes the system far more flexible with respect to the operations which it incorporates. And in general, the capabilities listed in Table II are a far from exhaustive list of all the capabilities of the MACSYMA system, which also incorporates a procedure for calculating a broad range of definite integrals, several special functions, direct and inverse Laplace transforms, and much more.[24,39]

TABLE II. Mathematical objects and operations on these objects which are incorporated in the various systems in Table I.

System →	SCHOONSCHIP	CLAM	REDUCE-2	SYMBAL	CAMAL	AVTO-ANALITIK	MACSYMA
Elementary functions	No	Most	Some	Few	Most	Most	All
Rational-fraction expressions	No	Yes	Yes	Yes	Yes	No	Yes
Search for largest common divisor	No	No	Yes	No	In simple cases	No	Yes
Differentiation	No	Yes	Yes	Yes	Yes	Yes	Yes
Integration	No	No	No	In simple cases	In simple cases	In simple cases	Yes
Complex quantities	Yes	No	Yes	Yes	Yes	No	Yes
Rational numbers	Yes	Yes	Yes	Yes	Yes	No	Yes
Arithmetic of floating-decimal numbers	Fast	No	Slow	No	Fast	Fast	Fast
Operation with fragments of power series	No	No	Fair	Excellent	Good	No	Excellent
Operation with fragments of Fourier series	No	No	No	Special kind	Excellent	No	Good
Vector and tensor algebra	Fair	Special kind	Good	Fair	Fair	No	Excellent
Metric algebra	No	No	Good	Fair	No	Fair	Excellent
Algebra of γ matrices and spinors	Excellent	No	Good	No	No	No	No
Noncommutative algebra	Good	No	Fair	No	No	Fair	Excellent

We turn now to several successful applications of analytic programming systems in physics and mathematics.

3. APPLICATIONS IN PHYSICS

a) Celestial mechanics

One of the fundamental problems of celestial mechanics is to construct an analytic theory of celestial objects, i.e., the major planets, their satellites, artificial satellites, space stations, etc. The term "theory" is understood in celestial mechanics as meaning a set of equations which determine the position of the celestial object as a function of the time. In order to construct this theory it is necessary to solve a system of differential equations describing the motion of the object, and only in trivial cases can a closed solution be found. Perturbation theory is the most general approach used in deriving analytic theories of celestial objects. Since in most cases the motion of the object is nearly periodic, the solution is usually sought as a series of trigonometric functions whose arguments are linear functions of the time; the coefficients of the series are polynomials in the small parameters of the particular problem.

These series are called Poisson series and have the general form

$$\varphi(x,\ y)=\sum_{j}\{P_j(x)\cos(j\cdot y)+Q_j(x)\sin(j\cdot y)\}, \tag{3.1}$$

where $x=\{x_1,x_2,\ldots,x_m\}$ is the vector of polynomial variables; $y=\{y_1,y_2,\ldots,y_n\}$ is the vector of trigonometric variables; $j=\{j_1,j_2,\ldots,j_n\}$ is a vector with integer components; and $P_j(x)$ and $q_j(x)$ are polynomials in the variables $x_1,x_2,\ldots,x_m$, which are indexed by the set $\{j_i\}$.

We can illustrate the situation with a simple example.[40] We consider the unperturbed Kepler motion described by the equation

$$E=u+e\sin E, \tag{3.2}$$

where e is the eccentricity of the orbit, E is the eccentric anomaly, in terms of which the coordinates and velocities of the elliptic motion are expressed in final form, and u is the mean anomaly, a linear function of

the time.

Treating e as a small parameter, as is usual in celestial mechanics, we seek a solution $E=f(u,e)$ of the Kepler equation in (3.2) by the method of successive approximations:

$$E=u+\lim_{n\to\infty}A_n, \tag{3.3}$$

where

$$\left.\begin{aligned} &A_0=0,\\ &A_1=e\sin u,\\ &\cdots\cdots\cdots\\ &A_{k+1}=\left[e\sin u\left(1-\frac{A_k^2}{2!}+\frac{A_k^4}{4!}-\ldots\right)+e\cos u\left(A_k-\frac{A_k^3}{2!}+\ldots\right)\right]_{k+1}. \end{aligned}\right\} \tag{3.4}$$

The index $k+1$ on the brackets in (3.4) means that terms of order higher than $k+1$ in e must be discarded. Transforming the right sides of (3.4) to expressions which are linear in the trigonometric functions, we find the following general expression for A_k:

$$A_k=\sum_{j=0}^{k}\left[P_j(e)\cos(ju)+Q_j(e)\sin(ju)\right]$$

with the polynomials P_j and Q_j. The solution in (3.3) of the Kepler equation in (3.2) is thus expressed in terms of the one-dimensional Poisson series in (3.1), in which the mean anomaly u is a trigonometric variable, and the eccentricity is a polynomial variable.

The high precision required of the calculations in celestial mechanics means that hundreds, thousands, or even tens of thousands of terms must be taken into account in the Poisson series and the associated series. It is thus not surprising that as early as 1958 (Ref. 41) astronomers were considering the use of computers for analytic calculations, and it was only a year later that the first program was published,[42] for an IBM 650, for analytic calculations of Poisson series. Since then, many programs have been written, and some analytic programming systems have in fact been developed especially for celestial mechanics.[40,43] Further motivation for the development of analytic programming systems for celestial mechanics came from the desire to check the results found back in the middle of the nineteenth century in a monumental work by Delaunay.[44] Delaunay derived an analytic theory for the motion of the moon in seventh order in the small quantities, such as the eccentricities $e\approx1/120$ and $e'\approx1/60$, the dimension ratio $a/a'\approx1/400$, and the sine of the inclination of the lunar and terrestrial orbits, $\gamma=\sin i\approx1/11$. A distinctive feature of the lunar motion is that the lunar orbit is outside the region in which the earth's attraction is stronger than the sun's. In constructing a theory for the moon it is thus necessary to consider the higher orders of perturbation theory. In calculating the perturbation function and then integrating the equations of motion of the moon, Delaunay used thousands of terms and ended up devoting 20 years to the project. In 1958, the informed estimate[41] was that 200 programming man-years would be required to reproduce Delaunay's results on a computer. Fortunately, this pessimistic estimate turned out to be very wrong, and in 1970 a group of three people[45], working for a year, reproduced Delaunay's results by means of the MAO system, developed by one of the group, A. Rom.[16] Remarkably, only a single error was found in Delaunay's multivolume work,[44] which contained about 40 000 equations.

The Poisson series in (3.1), the basic mathematical object of analytic perturbation theory, constitute a set which is closed with respect to the operations of addition, subtraction, and multiplication. Furthermore, if a suitable subset of the Poisson series is used, it is possible to introduce the operation of integration and to carry out several substitutions without leaving this subset. Another remarkable feature of (3.1) is that polynomial algebra and the standard operations on trigonometric (exponential) functions are sufficient for manipulating these series. Since in practice it is necessary to process a huge number of terms, it is clear that the most effective way to use computers is to develop special Poisson processors. The principal requirements which must be met by the corresponding algorithms are speed and a minimum demand on operational memory.

In the Soviet Union, this approach is being pursued actively in the Institute of Theoretical Astronomy. Using the AMS system which they had developed previously[17] for analytic operations with Poisson series, Brumberg and Isakovich[46] developed and adapted for the BÉSM-6 computer some FORTRAN Kepler-processor procedures and some procedures for exapnding the perturbation function in satellite problems. Other special FORTRAN systems have also been developed in the Institute of Theoretical Astronomy for manipulating both Poisson series[47] and power series.[48]

For a wide range of problems in celestial mechanics, especially those which require going beyond Poisson series (see, for example, the applications discussed in the review by Jeffrys[43]), more general analytic programming systems can of course be used to advantage. The CAMAL system,[18] developed for the general theory of relativity as well as for celestial mechanics, is particularly convenient. To illustrate the use of a universal system in celestial mechanics, we will cite the work by Anderson and Lau.[49] Using the MACSYMA system,[24] they calculated the temporal variations in the parameters of a Kepler orbit in first-order perturbation theory for an arbitrary perturbation function.

We recommend the review by Barton and Fitch[40] and that by Jeffrys[43] for a more detailed discussion of the range of celestial-mechanics problems which can be solved by analytic programming systems.

b) General theory of relativity

Over the past decade, much work has been carried out in the general theory of relativity by means of analytic programming systems. In many cases, these systems have played a leading role, since the extremely extensive calculations essentially rule out manual work. As an example of the most common calculations in the general theory of relativity—calculations which can be carried out completely on computers—we will consider the procedure for calculating the Riemann tensor (the curvature tensor) and the associated quanti-

ties for a given covariant metric $g_{ij}(i,j=0,1,2,3)$, specified by

$$ds^2 = g_{ij}\,dx^i\,dx^j.$$

To calculate the contravariant metric g^{ij} it is necessary to solve the system of linear equations

$$g^{il}g_{jk}=\delta^i_k, \tag{3.5}$$

i.e., to invert the matrix $\|g_{ij}\|$.

The Christoffel symbols of the first kind are defined by the first partial derivatives of g_{ij}:

$$\Gamma_{kij}=\frac{1}{2}\left(\frac{\partial g_{ik}}{\partial x^j}+\frac{\partial g_{jk}}{\partial x^i}-\frac{\partial g_{ij}}{\partial x^k}\right). \tag{3.6}$$

Suppressing the first index on Γ_{kij}, we find the Christoffel symbols of the second kind:

$$\Gamma^k_{ij}=g^{kp}\Gamma_{pij}. \tag{3.7}$$

The next step is to calculate the Riemann tensor from the equation

$$R_{ijkl}=\frac{\partial}{\partial x^k}\Gamma_{ijl}-\frac{\partial}{\partial x^l}\Gamma_{ijk}+\Gamma_{pil}\Gamma^p_{jk}-\Gamma_{pik}\Gamma^p_{jl}. \tag{3.8}$$

The convolution of the Riemann tensor yields the Ricci tensor,

$$R_{ij}=g^{pq}R_{piqj}, \tag{3.9}$$

and the scalar curvature,

$$R=g^{ij}R_{ij}. \tag{3.10}$$

These latter quantities appear, through the Einstein tensor

$$G_{ij}=R_{ij}-\frac{1}{2}Rg_{ij}, \tag{3.11}$$

in the basic equation of the general theory of relativity: the Einstein equation,

$$G_{ij}=T_{ij}, \tag{3.12}$$

where T_{ij} is the energy-momentum tensor.

Calculations from Eqs. (3.5)–(3.11) only require a matrix inversion and a differentiation in addition to the polynomial operations, and in principle they can be handled by most analytic programming systems. Special programs for such calculations were available in even the earliest systems, GRAD-ASSISTANT and FORMAC, and are described in Ref. 50. However, difficulties with the size of the operational memory of the IBM 7090/7094, which was used for these calculations, restricted the practical applications to problems in which, say, the components of the Ricci tensor contained no more than 100 terms.

To optimize calculations of the type in (3.5)–(3.11), and to generate some additional capabilities (working with tetrads, working with a metric containing a small parameter, etc.), several specialized analytic programming systems were developed for solving problems in the general theory of relativity. Some examples are ALAM[13] (and its modification LAM for the IBM/360 370 and its modification CLAM[14] for the CDC 6000/7000), CAMAL,[18] and SHEEP.[15] To illustrate the capabilities of these specialized analytic programming systems, we consider the metric proposed by Bondi *et al.*[51] for analyzing the gravitational radiation of a rotating star:

$$ds^2=\left[V\frac{\exp(2\beta)}{r}-U^2r^2\exp(2\gamma)\right]du^2+2\exp(2\beta)\,du\,dr + 2Ur^2\,du\,d\theta-r^2[\exp(2\gamma)\,d\theta^2-\exp(-2\gamma)\sin^2\theta\,d\varphi^2], \tag{3.13}$$

where U, V, β, and γ are functions of the coordinates u, r, and θ.

To calculate all the tensors for the metric in (3.13), up to the Einstein tensor in (3.11), by the CLAM system requires only 32 sec on a CDC 6500.

Most applications of analytic programming systems in the general theory of relativity involve analyzing the Einstein equation in (3.12) in vacuum ($T_{ij}=0$). In this case the condition $G_{ij}=0$ is equivalent to the condition

$$R_{ij}=0. \tag{3.14}$$

Because of the symmetry properties of the Ricci tensor and the metric g_{ij}, Eqs. (3.11) constitute a system of ten second-order partial differential equations for ten unknown functions g_{ij} $(i\geq j)$.

In 1959 Harrison[52] analyzed an important class of solutions of the vacuum equations in (3.14)—those having the structure

$$g_{ij}=A_i(x_0,x_1)\,B_j(x_0,x_2)\,\delta_{ij}. \tag{3.15}$$

Substitution of (3.15) into (3.14) reduces the latter to ordinary differential equations for the functions A_i and B_j. Using this approach, Harrison found all the different solutions of the type in (3.15); there turned out to be 40. Some of the solutions were simple, while others had a complicated algebraic structure. It was not possible to check the latter by direct substitution into (3.14) without using computers. In 1972, Harrison's solutions were checked independently by d'Inverno and Russell-Clark,[53] using the ALAM system, and by Fitch,[54] using the CAMAL system. As a result it was found that Harrison's expressions for g_{ij} did not satisfy Eqs. (3.14) in four cases out of the 40.

d'Inverno and Russell-Clark did not, however, stop after reaching this important conclusion.[53] They classified all of Harrison's solutions on the basis of the different types of metrics as proposed by Petrov.[55] In Petrov's classification, a metric of one type cannot be transformed into a metric of another type by means of a coordinate transformation, so the metrics of different types are different in an essential way. To determine which type a given metric was, d'Inverno and Russell-Clark[53] developed a special algorithm based on an analysis of the multiplicity of roots of the fourth-degree equation

$$\Phi_0Z^4+4\Phi_1Z^3+6\Phi_2Z^2+4\Phi_3Z+\Phi_4=0; \tag{3.16}$$

here the $\Phi_i(i=0-4)$ are the Newman–Penrose scalars, which are constructed from the Weyl tensor,

$$G_{ijkl}=R_{ijkl}+\frac{1}{2}(g_{il}R_{jk}-g_{ik}R_{ij}+g_{jk}R_{il}-g_{jl}R_{ik})-\frac{R}{6}(g_{il}g_{kj}-g_{ik}g_{ij}).$$

We note that the Weyl tensor reduces in vacuum to the Riemann tensor by virtue of (3.14). To determine the multiplicity of roots of Eq. (3.16) and then determine the type of matrix, it is not absolutely necessary to

solve this equation. It is sufficient to check to see whether a certain set of polynomial equations involving the function Φ_i holds. These equations are quite complicated, and some contain terms of sixth degree in the variables Φ_i. Nevertheless, the effort to find a Petrov classification of all of Harrison's solutions was successful thanks to analytic programming systems, specifically, the ALAM system.

These examples, of course, fall far short of exhausting the complete list of applications of analytic programming systems in the general theory of relativity. Several other applications can be found in, for example, Refs. 56 and 57 and in the reviews of Refs. 40, 58, and 59. Among the systems used at the Joint Institute for Nuclear Research (Table I), CLAM and CAMAL are devoted to calculations in the general theory of relativity. This does not mean, however, that other systems (in particular, SYMBAL and REDUCE-2) are not useful in certain cases. As an example of the problems of this type, we can cite the calculation of the tidal forces in the vicinity of a black hole which were carried out[60] by the REDUCE-2 system.

c) Quantum field theory

One of the most successful applications of analytic programming systems, along with celestial mechanics and the general theory of relativity, has been in perturbation-theory calculations in quantum field theory. This important field of theoretical physics combines exceedingly complicated calculations with a comparatively small number of necessary mathematical operations. Computers, especially when equipped with analytic programming systems, have found many applications (see, for example, the reviews in Refs. 40 and 61–65) in each step of the calculations by the well-known Feynman-diagram technique.[66–68] Let us consider these steps in succession.

I. Generation of diagrams corresponding to the given order of perturbation theory for a given process. Four different programs have been published for generating Feynman diagrams in quantum field theory.[69–72] The best-developed is the program worked out by Sasaki[72] especially for quantum electrodynamics. This program, written in LISP (and available for the CDC 6500 at the Joint Institute for Nuclear Research), generates only those diagrams which are topologically nonequivalent, rejecting those of no physical interest (those which are not connected, those which contain closed electron loops with an odd number of vertices, etc.).

II. Derivation of integrands. For those models of quantum field theory which describe particles with a nonzero spin, this step of the calculation is frequently exceedingly laborious. For example, in quantum electrodynamics it is necessary to carry out some laborious operations of the algebra of Dirac γ matrices, in particular, to simplify expressions of the type $\sum_\mu \gamma_\mu \ldots \gamma_\mu$ and to calculate the trace. The effective execution of these operations on a computer required the development of special algorithms,[73,74] which are now incorporated in the SCHOONSCHIP,[7] ASHMEDAI,[12] and REDUCE-2 (Ref. 10) systems. Each of these three systems allows all the calculations involved in constructing the integrands to be carried to completion. In higher-order perturbation theory, however, SCHOONSCHIP and ASHMEDAI are preferred; these systems, which were developed especially for quantum field theory, demand much less of the computer than does the universal REDUCE-2 systems.

III. Elimination of divergences. In the higher orders of perturbation theory, the calculations in step II are known to lead to divergent integrals, which pose one of the main problems in quantum field theory.[66–68] A rigorous mathematical procedure for assigning meaning to these integrals, namely the Bogolyubov–Parasyuk R operation,[66,75] allows the divergent parts of these integrals to be seprated out, so that renormalized integrands corresponding to finite integrals can be constructed. Unfortunately, and despite a few important results,[76] a general algorithmic approach has not yet been developed for the extremely laborious renormalization procedure in quantum field theory. For this reason, in quantum electrodynamics, for example, only part of the R operation has been adapted for computer calculations, namely, the procedure of analyzing the structure of the divergences of a given diagram.[77] At present, the computer calculation procedure can be carried out completely for only scalar theories. Calmet and Perrottet,[70] for example, have examined the simplest case of the scalar theory: the super-renormalizable model $g\varphi^3$, in which the divergences arise only in second-order diagrams for the self-energy. Calmet and Perrottet wrote a LISP program for constructing the renormalized amplitudes in the $g\varphi^3$ model. A much more general approach was developed by Tarasov in Ref. 78, where an algorithm was given, and a SCHOONSCHIP program was described which implemented the R operation completely for scalar theories for the case of an arbitrary subtraction point.

IV. Integration. In by no means all cases, of course, is it possible to calculate the renormalized Feynman integrals analytically; these a special class of multiple integrals. In several important cases, nevertheless, in which the procedure for the analytic integration is known, analytic programming systems have had much success. These cases are summarized in Table III, which is taken from Ref. 65, to which the reader is referred for further details. It might appear at first glance that for those problems in which we are not interested in the exact dependence of the Feynman amplitude on any parameters (the kinematic variables) an analytic calculation would not be necessary. An ordinary numerical integration would suffice. However, because of the structure of the integrand, which usually has integrable singularities, the only reliable numerical-integration method is the Monte Carlo method.[79] The Monte Carlo method, however, is very slow and requires much computer time. For example, to evaluate the septuple parametric integrals which arise in sixth-order perturbation theory for the anomalous magnetic moment of the electron within 1% requires several hours on the most powerful computers in existence. Analytic programming systems, in contrast, yield an accurate answer in ~10 min (Table III). Even in those

TABLE III.

Representation of integral	Order of perturbation theory	Number of integration loops	Multiplicity of integral	System	Computer	Typical calculation time	Quantity calculated	Reference
Momentum*	6	3	12	ASHMEDAI	UNIVAC-1108	10 min	Anomalous magnetic moment of electron	80
Momentum*	6	2	8	MACSYMA	DEC PDP-10	2 min	Divergent part of the electric-charge renormalization constant	81
Ditto	4	2	8	SCHOONSCHIP	CDC-6600	–	Electron form factor	82
"	6	3	12	SCHOONSCHIP	CDC-7600	–	Anomalous magnetic moment of electron and muon	83
Parametric**	4	2	4	SCHOONSCHIP	CDC-6500	1 min	Anomalous magnetic moment of electron, Lamb shift	84
Parametric**	4	2	4	REDUCE-2	DEC PDP-10	2 min	Anomalous magnetic moment of electron, Lamb shift	85

*The integral is calculated directly in momentum space.
**The integral is represented in terms of Feynman parameters.

cases in which the analytic evaluation of the integral cannot be carried out completely, the use of analytic programming systems to evaluate some of the repeated integrals greatly improves the accuracy of numerical calculations.

V. Calculation of scattering cross sections. Many problems of high-energy physics require calculations of differential cross sections. Calculations of this sort require an extension of the calculation scheme described above. All the calculations which involve squaring a Feynman amplitude (matrix element) are similar to the calculations of step II and can be carried out completely on a computer by means of the SCHOONSCHIP, ASHMEDAI, and REDUCE-2 systems. Much more difficult is evaluating integrals over a phase space, in which case the integrand typically depends on parameters (the masses and the kinematic variables), and the integration limits have a complicated structure. Under these conditions, the analytic programming systems may be exceptionally useful. As an example, we can cite the work by Bardin *et al.*,[86] who used the SCHOONSCHIP system to derive the exact electromagnetic correction of lowest order to the elastic scattering of two spin-1/2 particles and the elastic scattering of a spin-0 particle by a spin-1/2 particle. The purpose of those calculations was to analyze some experiments in which elastic reactions were not discreiminated from reactions involving the emission of a photon by bremsstrahlung. Another interesting example is the calculation[87] of the differential distribution (the ratio of the differential cross section to the total cross section) of $e^-\mu^+$ pairs obtained in the lepton decay of a pair of heavy leptons created in e^+e^- annihilation. Assuming the $V-A$ weak interaction, Linke *et al.*[87] calculated the distribution of the $e^-\mu^+$ lepton pair in the lowest order of perturbation theory, using SCHOONSCHIP to calculate the square of the matrix element for the process and for a partial integration over the phase space of the undetected particles (four neutrinos). The rest of the phase integral was evaluated numerically. The analytic part of the calculation required 7 min on an CDC Cyber-175 computer; the maximum length of the intermediate expressions was ≈40 000 terms.

We turn now to the greatest success of analytic programming systems in quantum field theory: the calculation, in sixth-order perturbation theory, of the anomalous magnetic moment of the electron, clearly a quantity of fundamental importance in physics. This moment is known[66,67] to be that part of the total magnet moment

$$\mu = \mu_0 (1 + a),$$

which results from the radiative corrections to the point electromagnetic vertex corresponding to a magnetic moment μ_0, the Bohr magneton,

$$\mu_0 = \frac{e}{2m}.$$

The quantity a can be written as a perturbation series in powers of $\alpha/\pi = e^2/4\pi^2$:

$$a = a_2\left(\frac{\alpha}{\pi}\right) + a_4\left(\frac{\alpha}{\pi}\right)^2 + a_6\left(\frac{\alpha}{\pi}\right)^3 + a_8\left(\frac{\alpha}{\pi}\right)^4 + \ldots .$$

The coefficient a_2, which is governed by a single Feynman diagram (Fig. 2a), is

$$a_2 = \frac{1}{2}$$

and was calculated by Schwinger[89] in 1948.

The calculations in the next order of perturbation theory require five different two-loop diagrams (Fig. 2b). Their total contribution to a was calculated ten years after Schwinger's work by Peterman[90] and Sommerfeld[91]:

$$a_4 = \frac{197}{144} + \frac{\pi^2}{12} - \pi^2 \ln 2 + \frac{3}{4}\zeta(3),$$

where

$$\zeta(3) = \sum_{k=1}^{\infty} \frac{1}{k^3} = 1.2020569\ldots .$$

To determine the coefficient a_6, i.e., the contribution from sixth-order perturbation theory, it becomes nec-

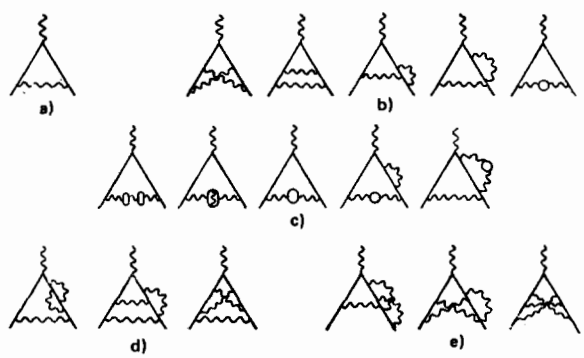

FIG. 2. One-loop, two-loop, and certain three-loop vertex diagrams for the anomalous magnetic moment of the electron.

TABLE IV. Experimental and theoretical progress on the anomalous magnetic moment of the electron over the past 5 years.

Year	Calculated 7th-order diagrams: Analytically	Calculated 7th-order diagrams: Numerically	a_6^{theo}	a_6^{expt}	a^{theo}	a^{expt}	$\Delta a^{expt}/a^{expt}$	$\Delta\mu^{expt}/\mu^{expt}$
1973 [95]	5	33	1.21 ± 0.07	1.060 ± 0.33	$(1159651.9\pm 2.5)\cdot 10^{-9}$	$(1159656.7\pm 3.5)\cdot 10^{-9}$	$3.0\cdot 10^{-6}$	$3.5\cdot 10^{-9}$
1978 [94]	30	10	1.184 ± 0.007	1.189 ± 0.025	$(1159652.375\pm 261)\cdot 10^{-9}$	$(1159652.41\pm 0.20)\cdot 10^{-9}$	$1.7\cdot 10^{-7}$	2.0×10^{-10}

essary to calculate 40 different three-loop diagrams, and these calculations are so long as to be essentially impossible without computers. Suffice it to say that not a single three-loop vertex diagram has been calculated analytically without the use of analytic programming systems.[92-94] However, while in 1973 it was possible to calculate analytically only five of the diagrams (those in Fig. 2c; and all were calculated with the help of the SCHOONSCHIP system), over the past five years the use of the SCHOONSCHIP[7] and, especially, ASHMEDAI [12] systems has resulted in the analytic calculation of another 25 diagrams. The net result has been to reduce by an order of magnitude the error in the theoretical value of a_6 (Table IV). The diagrams which are most difficult for the calculations are those in Fig. 2d; in the intermediate stage of the corresponding calculations, there are as many as 24 000 terms, and each diagram required[80] about 150 min on a UNIVAC 1108 (this time includes steps II–IV of the calculation procedure described above, which is carried out completely in the ASHMEDAI system).

For the other ten diagrams contributing to a_6, calculation steps II and III (the construction of the integrands and the elimination of the divergences) were carried out by analytic programming systems, and the rest of the calculations (step III, the integration) were carried out numerically.

The calculated results (Table IV) turned out to be in extremely gratifying agreement with recent precise measurements[96] of the anomalous magnetic moment of the electron, within a relative error of $2\cdot 10^{-10}$.

In the next few years we can expect further progress in the refinement of both experimental[96] and theoretical[94] data. For example, there is every reason to believe that of the ten seventh-order diagrams which have been found numerically at the present time, seven will soon be calculated analytically by known methods. At this point it is not clear how to calculate the three remaining nonplanar diagrams (Fig. 2e), for which the calculation technique developed by Levine *et al.*[80] is not applicable. If, however, we are interested in only their numerical values, we can bring the error in the theoretical value of a_6 to[97]

$$0.13\left(\frac{\alpha}{\pi}\right)^4 \approx 4\cdot 10^{-12}.$$

At this accuracy level, however, we should take into account other contributions to a, mainly the muon loop,[94]

$$a\,(\text{muon loop}) = 2.8\cdot 10^{-12} \approx 0.1\left(\frac{\alpha}{\pi}\right)^4,$$

the hadron vacuum polarization,

$$a\,(\text{hadron}) = 2\cdot 10^{-12} \approx 0.07\left(\frac{\alpha}{\pi}\right)^4,$$

and weak interactions

$$a\,(\text{Salam}-\text{Weinberg}) = 0.05\cdot 10^{-12} \approx 0.0017\left(\frac{\alpha}{\pi}\right)^4.$$

The fact that these effects are small in comparison with $(\alpha/\pi)^4$ means that the next step in refining the theoretical value of the anomalous magnetic moment of the electron will be to go to eighth-order perturbation-theory calculations in quantum electrodynamics.

For this purpose it is necessary to calculate the contribution of 430 different four-loop vertex diagrams. Of them, 161 are found by inserting electron loops in a lower-order diagram, and the corresponding calculations are no more complicated than those for diagrams making the contribution $\sim(\alpha/\pi)^3$. The other 269 diagrams correspond to decuple integrals in a parameter space, for which the integrand is 10–20 times as long as that for three-loop diagrams.

Work has already been begun[97] on a_8. In particular, a SCHOONSCHIP program has been written for constructing integrands. This program requires from 2 to 10 min per diagram on a CDC 7600. According to estimates by Kinoshita,[97] these calculations would require several hundred hours for a 10% error if the integration were carried out by numerical methods on the CDC 7600. It is believed that the total calculation effort required in this project will be from 4 to 6 man-years.

In recent years, analytic programming systems have found some interesting applications in non-Abelian gauge models of quantum field theory. In particular, we can cite the use of the SCHOONSCHIP system for two-loop calculations[98,99] of the Gell-Mann–Lau function and the anomalous propagator dimensionalities in the Yang-Mills theory with an arbitrary gauge parameter. These calculations were carried out in the JINR on a CDC 6400; they required 84 min of computer time for the total of 33 two-loop diagrams. Another important example would be the REDUCE-2 calculation of the violation of the Okubo-Iizuki–Zweig rules in lowest-order perturbation theory in quantum chromodynamics which results from the radiative decay of heavy pseudoscalar particles.[100]

d) Plasma physics

Analytic programming systems have had much success in this exceedingly important field of physics.[101-106] The most interesting applications have been in analyzing the hydrodynamic condition for plasma stability in tokamaks.[101-103] The corresponding numerical calculations would require several hours of computer time, even on the most powerful computers in existence, and even then the conclusion reached about the plasma stability would be just Barely reliable. In contrast, purely analytic calculations generate expressions in the intermediate steps which are so long that the user runs into problems with the computer memory. Further

more, these analytic calculations also require much computer time. The best approach here has been to combine numerical and analytic methods.[102]

Let us examine the structure of the calculations which have been carried out on the plasma-stability problem. When a displacement ξ occurs in the plasma, the plasma stability is governed by the deviation (δU) of the potential energy of the plasma from its equilibrium value. If $\delta U < 0$, the plasma is unstable. According to the MHD theory, δU can be written as the following triple integral, which depends on the geometry of the particular device and the nature of the displacement ξ:

$$\delta U = \frac{1}{2} \int\limits_{\substack{\text{plasma}\\ \text{volume}}} d\nu \{g_{ij} Q^i Q^j + \xi [\mathbf{QJ}] + \phi\}, \tag{3.17}$$

where

$$\mathbf{Q} = \nabla \times [\xi \mathbf{B}],$$

$$\phi = (\nabla \cdot \xi)(\xi \nabla p) + \frac{5}{3} p (\nabla \cdot \xi)^2,$$

B and J are the magnetic field and current density, respectively, in the equilibrium state, g_{ij} is the metric tensor, and p is the plasma pressure.

If $\phi = 0$, then the displacement ξ satisfies

$$\nabla \cdot \xi = 0.$$

The problem is to evaluate the integral in (3.17) with the necessary accuracy. For the calculations in Ref. 102, which we are discussing here, a curvilinear coordinate system X^1, X^2, X^3 reflecting the axial symmetry of a tokamak was selected (Fig. 3):

$$\left.\begin{aligned} X^1 &= \sqrt{\left[z^2 (r^2 - \gamma) + \frac{\alpha^2}{4} (r^2 - R^2)^2\right] \frac{1}{\alpha^2}}, \\ X^2 &= \varphi, \\ X^3 &= \operatorname{Arctg}\left(\frac{2}{\alpha} \frac{z\sqrt{r^2 - \gamma}}{r^2 - R^2}\right). \end{aligned}\right\} \tag{3.18}$$

Here z, φ, r are the cylindrical coordinates; the constants α, γ, and R specify the geometry of the device.

In system (3.18), $X^2, X^3 \in [0, 2\pi]$, $X^1 = \text{const}$ is a magnetic surface, and $X^1 = X^1_b = \text{const}$ corresponds to the plasma boundary.

The variation in the plasma potential energy, δU, is found by expanding ξ in a Fourier series:

$$\xi = \sum_{k, m} \xi_{k, m} (X^1) \exp(ikX^2 + imX^3). \tag{3.19}$$

Because of the axial symmetry of the problem, we can seek the variation δU separately for each value of k. The following displacement was studied in Ref. 102:

$$\xi_k = \sum_m \xi_{k, m} (X^1) \exp(ikX^2 + imX^3) K_0 (X^1, X^3),$$

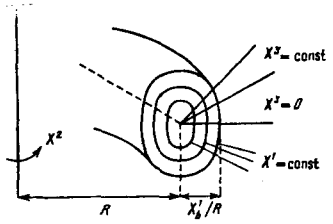

FIG. 3. Curvilinear coordinate system used in analyzing the plasma stability in a tokamak.

where

$$K_0 (X^1, X^3) = \sum_{l=0}^{L} d_l \left(\frac{2X^1}{R^2} \cos X^3\right)^l. \tag{3.20}$$

Two types of functions $\xi^i_{k, m}(X^1)$ were chosen for the variation: I) polynomials in X^1; II) Bessel functions of integer order. Equation (3.17) is studied by varying the function (3.20) with respect to its parameters and by varying the functions $\xi^i_{k, m}(X^1)$ with respect to their parameters. In case I, the integration over all the X^i can be carried out analytically; in case II, Eq. (3.17) reduces to elliptic integrals after an analytic integration over X^2 and X^3, and these elliptic integrals must be evaluated numerically. If, however, the integrand is expanded in a Taylor series, then a completely analytic result can be found for each term in the expansion.

In practice, the calculations were carried out[102] both through the use of a Taylor series (21 terms in the series were required for the accuracy desired) and without this series, but with a numerical integration over X^1. The REDUCE-2 system was used for the analytic calculations,[10] carried out as follows:

1. Calculation of the quantities Q^i and then of the integrand in (3.17).
2. Integration over X^2.
3. Preparation for an integration over X^3.

3.1. Expansion of the integrand in a Taylor series (if necessary).

3.2. Introduction of the function K_0 in accordance with (3.20).

4. Integration over X^3.
5. Integration over X^1. In the case in which this integral is evaluated numerically, a special transformation of the integrand is carried out to simplify the subsequent calculations.
6. Determination of the final expression by simplifying the results of the preceding calculations.

In the case in which the functions $\xi_{k, m}(X^1)$ are represented in terms of Bessel functions of integer order, the analytic part of the calculations required 40 min on an Amdahl-470V/6 computer; then the numerical calculations required about 50 sec.

According to the MHD theory, the integral in (3.17) is the difference between two large numbers which are approximately equal in magnitude:

$$\delta U = \delta U_+ - \delta U_- \qquad (\delta U_\pm > 0).$$

Consequently, $\delta U_\pm$ must be calculated very accurately. The calculations carried out by the method described above showed that the numerical calculations were not sufficiently accurate but that combined numerical and analytic methods were quite accurate; in principle, any accuracy level could be achieved by using a sufficient number of terms in the series expansion.

e) Hydrodynamics

In hydrodynamics, and for that matter in many other fields of applied mathematics, it is necessary to solve some complicated systems of partial differential equations. Attempts to solve these equations by analytic methods run into insurmountable difficulties, except in

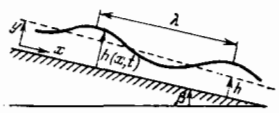

FIG. 4. Geometry of the problem of the flow of a thin, two-dimensional liquid down an inclined plane.

comparatively simple cases. However, even when an analytic solution method has been developed for some particular hydrodynamic problem, the implementation of this method generally requires some tedious analytic calculations. In such situations, analytic programming systems are extremely useful, and frequently they are the only solution method available.

As an example, we consider the problem of constructing the evolution equation for the function $h(x,t)$ which describes the free surface of a thin, two-dimensional flow of a very viscous liquid down an inclined plane (Fig. 4).

This problem has been solved successfully by means of the REDUCE-2 system[107] in the approximation of long surface waves. Here we will follow Ref. 107 and give an exact mathematical formulation of the problem and then briefly describe the solution method.

This type of liquid motion has two scale dimensions: the length of the surface wave, λ, along the x axis and the thickness of the liquid layer, h, along the y axis.

We introduce the stream function $\psi(x,y,t)$, which in our two-dimensional case allows us to reduce the Navier–Stokes equations to a single partial differential equation. We transform to dimensionless variables by using the scale transformations

$$(x,\ y,\ t)\to\left(\frac{\alpha x}{h_0},\ \frac{y}{h_0},\ \frac{\alpha t\bar{u}_0}{h_0}\right),$$
$$(\psi,\ p,\ h)\to\left(\frac{\psi h_0}{\bar{u}_0},\ \frac{ph_0}{\mu\bar{u}_0},\ \frac{h}{h_0}\right),$$

where $h_0=ah/k$ (a and k are the heat-transfer coefficient and the thermal conductivity, respectively) is the Nusselt thickness for this flow, $\bar{u}_0$ is the average velocity, $\alpha=2\pi h_0/\lambda$ is the dimensionless wave number, and μ is the viscosity.

Then the basic nonlinear dynamic equation for $\psi(x,y,t)$ becomes

$$\psi_{yyyy}=\alpha\,\mathrm{Re}\,(\psi_{tyy}+\psi_y\psi_{xyy}-\psi_x\psi_{yyy})-2\alpha^2\psi_{xxyy}+\alpha^3\,\mathrm{Re}\,(\psi_{txx}-\psi_x\psi_{xxy}+\psi_y\psi_{xxx}-\alpha^4\psi_{xxxx}),\quad(3.21)$$

where Re is the Reynolds number. The stream function must satisfy, along with Eq. (3.21), the following boundary conditions:

1. The adhesion conditions:

$$\psi_x=0\quad\text{at}\quad y=0,\qquad(3.22)$$
$$\psi_y=0\quad\text{at}\quad y=0.\qquad(3.23)$$

2. The surface-tension conditions:

$$(\psi_{yy}-\alpha^2\psi_{xx})(1-\alpha^2h_x^2)-4\alpha^2h_x\psi_{xy}=0\quad\text{at}\quad y=h(x,t),\qquad(3.24)$$
$$\psi_{yyy}+3+\alpha^2\psi_{xxy}-\alpha\mathrm{Re}\,(\psi_{ty}+\psi_y\psi_{xy}-\psi_x\psi_{yy})-\alpha^2h_x\psi_{yyx}-3\alpha h_x\mathrm{ctg}\,\beta-\alpha^4h_x\psi_{xxx}+\alpha^3\mathrm{Re}\,h_x(\psi_{tx}+\psi_y\psi_{xx}-\psi_x\psi_{yx})+2\alpha^2(\psi_{yxx}+\psi_{yyx}h_x)+8\alpha^4\psi_{xy}(h_x^2h_{xx}+2\alpha^2h_x^3h_{xx}+3\alpha^4h_x^5h_{xx})+\alpha\mathrm{Re}P\Big[h_{xxx}+3\alpha^2\Big(-\frac{1}{2}h_x^3h_{xxx}-h_{xx}^2h_x\Big)+15\alpha^4\Big(\frac{1}{8}h_{xxx}h_x^4+\frac{1}{2}h_{xx}^2h_x^3\Big)+35\alpha^6\Big(-\frac{1}{16}h_x^5h_{xxx}-\frac{3}{8}h_x^5h_{xx}^2\Big)\Big]+4\alpha^4h_x^2(\psi_{yxx}+\psi_{xyy}h_x)(1+\alpha^2h_x^2+\alpha^4h_x^4)+O(\alpha^9)=0\quad\text{at}\quad y=h(x,t).\qquad(3.25)$$

Here $P=\alpha^2W$, where $W=\sigma/h_0\bar{u}_0^2\rho$ is the Weber number (σ is the surface-tension coefficient, and ρ is the density), and β is the inclination of the plane (Fig. 3).

3. The kinematic condition at the free surface:

$$h_t+h_x\psi_y+\psi_x=0\quad\text{at}\quad y=h(x,t).\qquad(3.26)$$

Conditions (3.22)–(3.25), along with Eq. (3.21) are sufficient conditions for expressing ψ in terms of the function $h(x,t)$ and its derivatives; then condition (3.26) gives us an evolution equation for the function $h(x,t)$, which is the ultimate goal of this analysis.

In the longwave approximation ($\lambda\gg h$), the condition $\alpha\ll1$ holds, and we can seek a solution of problem (3.21)–(3.25) as the series

$$\psi=\sum_{k=0}^{\infty}\alpha^k\psi^{(k)}$$

by the method of successive approximations. For $\psi^{(0)}$ we have

$$\psi^{(0)}_{yyyy}=0,\qquad(3.27)$$

and at $y=h(x,t)$ we have

$$\psi^{(0)}_{yyy}+3=0,\quad\psi^{(0)}_{yy}=0,\quad\psi^{(0)}_y=0,\quad\psi^{(0)}_x=0.\qquad(3.28)$$

The problem in (3.27), (3.28) is easily solved:

$$\psi^{(0)}=\frac{3}{2}\left(y^2h-\frac{1}{3}y^3\right).$$

The rest of the process reduces to solving the equation

$$\psi^{(n)}_{yyyy}=f(\psi^{(n-1)},\ \psi^{(n-2)},\ \ldots,\ \psi^{(0)}),\qquad(3.29)$$

which has the structure

$$\psi^{(n)}_{yyyy}=\sum_{i=0}^{t_n}\bar{g}_i^{(n)}y^i\qquad(k=0,\ 1,\ \ldots),$$
$$\{t_n\}=\{0,\ 1,\ 5,\ 9,\ \ldots,\ t_{n-1}+4\},$$

where the $g_i^{(n)}$ depend on x and t through the function $h(x,t)$ and its derivatives, i.e.,

$$\bar{g}_i^{(n)}=\bar{g}_i^{(n)}(h,\ h_t,\ h_x,\ h_{xt},\ \ldots).$$

Then Eq. (3.29) is integrated easily:

$$\psi^{(n)}=\sum_{i=0}^{t_n}\frac{g_i^{(n)}y^{i+4}}{(i+4)(i+3)(i+2)(i+1)}+C_3^{(n)}y^3+C_2^{(n)}y^2+C_1^{(n)}y+C_0^{(n)}.\qquad(3.30)$$

Since the stream function is specified within an additive constant, conditions (3.22) and (3.23) can be put in the form

$$D^{(0)}\psi^{(n)}\equiv\psi^{(n)}(0)=0,\quad D^{(1)}\psi^{(n)}\equiv\psi^{(n)}_y(0)=0,$$

from which it follows immediately that

$$C_0^{(n)}=C_1^{(n)}=0\qquad(n=0,\ 1,\ 2,\ \ldots).$$

Then the quantities $\bar{g}_i^{(n)}$, $C_2^{(n)}$, $C_3^{(n)}$ are expressed in terms of the results of the preceding approximation by substituting (3.30) into (3.24) and (3.25), which can be written

$$D^{(3)}\psi^{(n)}\equiv\psi^{(n)}_{yyy}-f_3(\psi^{(n-1)},\ \psi^{(n-2)},\ \ldots,\ \psi^{(0)})=0,$$
$$D^{(2)}\psi^{(n)}\equiv\psi^{(n)}_{yy}-f_2(\psi^{(n-1)},\ \psi^{(n-2)},\ \ldots,\ \psi^{(0)})=0.$$

As a result, the kinematic condition in (3.26) can be transformed into the parabolic equation

$$\frac{\partial h}{\partial t}+[\psi^{(0)}(h,\ h)+\alpha\psi^{(1)}(h,\ h)+\alpha^2\psi^{(2)}(h,\ h)+\ldots]_x=0,$$

which describes the wavelike behavior of the free sur-

face of the liquid in this hydrodynamic problem.

The method described above reduces to polynomial operations and substitutions, so that it can be implemented by any analytic programming system in Table I.

Atherton and Homsy[107] reported the results calculated for $\psi^{(2)}(h,h)$ and $\psi^{(3)}(h,h)$ by REDUCE-2 [it is a comparatively simple matter to calculate the function $\psi^{(1)}(h,h)$ manually]. These results are quite complicated expressions. For example, $\psi^{(3)}(h,h)$ is a polynomial in the derivatives of the function $h(x,t)$, consisting of 60 terms of the type

$$\psi^{(3)}(h,h) = -\frac{83}{168}P^2\mathrm{Re}^3h^7h_{xxxx}h_{xxx} - \frac{3479}{1440}\mathrm{Re}^2\operatorname{ctg}\beta h^9h_{1xx} + \dots .$$

Other applications of analytic programming systems in hydrodynamics can be found, for example, in Refs. 108–110. Some interesting applications have also been found in aerodynamics. In particular, Cohen *et al.*[111,112] have recently used the REDUCE-2 system to calculate the parameters of a rapidly rotating ideal gas with a low Mach number.

f) Atomic and molecular physics and quantum chemistry

Numerical calculations have of course become an integral part of much research in atomic and molecular physics and also quantum chemistry. More recently, analytic programming systems have also found several applications.

For quantum chemistry, for example, these analytic systems constitute a powerful tool for standard analytic manipulations involving calculating the matrix elements of the product of creation and annihilation operators. The basic calculation difficulty here lies in reducing this product (which frequently contains many operators) to normal form by means of commutation relations. The best analytic programming system for analytic transformations of this type is the SCHOONSCHIP system.[7] Nerbrant[113] has recently used this system to develop a special program for rapid and efficient calculations of the matrix elements of the product of creation and annihilation operators for fermion fields.

The FORMAC system[20] has found some interesting applications in atomic and molecular physics. For example, Benesch[114] used FORMAC to calculate the radial distribution functions in coordinate and momentum space for helium and helium-like ions (Li^+, B^{3+}, O^{6+}, Ne^{8+}, Mg^{10+}).

Many calculations in atomic physics require knowledge of the Clebsch-Gordan coefficients. A FORMAC program has been written for analytic calculation of these coefficients; this program uses the familiar formula[115]

$$\begin{aligned}\langle j_1 j_2 m_1 m_2 | JM\rangle &= \sqrt{2J+1}\sqrt{\Delta(j_1,j_2,J)}\sqrt{(j_1+m_1)(j_1-m_1)(j_2+m_2)(j_2-m_2)(J+M)(J-M)}\\ &\times\sum_k(-1)^k[k!(J-j_2+k+m_1)!(J-j_1+k-m_2)!\\ &\times(j_1-j_2+J-k)!(j_1-k-m_1)!(j_2-k+m_2)!]^{-1},\end{aligned} \tag{3.31}$$

where

$$\Delta(j_1, j_2, J) = \frac{(j_1+j_2-J)!\,(j_2+J-j_1)!\,(J+j_1-j_2)!}{(j_1+j_2+J+1)!},$$

and the summation is carried out over all integers k for the numbers in the factorials are nonnegative. The program of Ref. 115 works from Eq. (3.31) to yield, for a given number j_2, an analytic expression for the Clebsch-Gordan coefficients as functions of the other parameters (j_1, m_1, m_2, J, M). This program requires, for example, 11 sec on the IBM 370/168 for $j_2 = 1/2$, 3/2, and 6; and 6 min for $j_2 = 2, 5/2, 3, 7/2, 4, 9/2$ (Ref. 2). There is an interesting applications of REDUCE-2 (Ref. 10) and SYMBAL[22] in Ref. 117, where five terms in the expansion of the energy in smooth perturbations of linear potentials in the Schrödinger equation are calculated.

In the quantum mechanics of atomic systems, a fundamental role is played by the Slater integrals,[118]:

$$\int_0^\infty\int_0^\infty P(n_1e_1;\,s)\,P(n_2e_2;\,r)\left\langle\frac{r}{s}\right\rangle^{k+1}\frac{1}{rs}P(n_3e_3;\,s)\,P(n_4e_4;\,r)\,ds\,dr, \tag{3.32}$$

where k is an integer, $\langle r/s\rangle = \min\{r/s, s/r\}$, and the functions $P(ne;r)$ are of the type $e^{-\alpha r}f(r)$, where $f(r)$ is a polynomial in r.

Analytic programming systems have proved exceptionally useful in evaluating the integrals in (3.32). We will mention, in particular, the use of ALTRAN[21] by Fischer and Prentice[119] and the use of FORMAC by Golden.[120]

4. APPLICATIONS IN MATHEMATICS

a) Evaluation of indefinite integrals

While differentiation is easily embodied in an algorithm, and was in fact the first mathematical operation to be carried out analytically on a computer,[2] the inverse operation—integration—presents incomparably greater difficulties. The first successful attempt in this direction was that by Slagle.[121] Soon after the appearance of LISP Slagle used it to write the program SAINT (Symbolic Automatic Integrator) which used a table of integrals and attempted to reduce the given integral to one of the tabulated integrals by one of the standard approaches which would be included in a university course in mathematical analysis.

After a few years, this heuristic approach was refined by Moses into the SIN program[122] (Symbolic Integrator); this program (along with the Risch algorithm[123] which it incorporates) is a constituent part of the MACSYMA [24] and SCRATCHPAD[25] systems.

The most important accomplishment, however, consisted of the development and dissemination of an algorithmic approach to calculating indefinite integrals. Special algorithms were developed for integrating rational functions,[125,126] and a universal algorithm for integrating elementary functions was discovered by Risch[123,124] (see also the reviews in Refs. 40 and 127).

The algorithmic approach, in contrast with the heuristic approach, finds the integral without making use

of any tables.

It should be noted that the advent of an algorithmic approach did not make the heuristic method obsolete; for a broad range of integrals, the heuristic method requires much less computer time. Furthermore, because of serious technical difficulties, the Risch algorithm has not yet been adapted to computer calculations in the case in which the integrand contains algebraic functions (expressions with noninteger powers).

Accordingly, the most powerful integration tools in modern analytic programming systems, such as the SIN program and the even more effective integration program which has recently been developed and disseminated in a new version of the REDUCE-2 system,[129] make fundamental use of both these approaches.

The SIN program, for example, implements the following strategy for evaluating an integral.[127]

Step I. An attempt is made to write the integral in the form

$$C\int f[u(x)]\,u'(x)\,dx,$$

where C is a constant, $u(x)$ is some function, $u'(x)$ is its derivative and $f(u)$ is a simple elementary function such as $\sin u$, $\cos u$, u^a, e^u, $\ln u$, or $\sin^{-1}u$ or a sum of such functions. If this can be done, then the integral can be evaluated by substituting in standard tabulated integrals. Otherwise, the calculation proceeds to the next step.

Step II. Depending on the structure of the integrand, 11 standard approaches are used; for example, using

1) the substitution $t=\tan(x/2)$ for trigonometric functions;

2) the Chebyshev method for integrals of the type

$$\int x^r (C_1 + C_2 x^q)^p\,dx,$$

where p, q, and r are rational numbers;

3) the Ostrogradskii–Hermite method for rational functions; etc.

If the integral still cannot be evaluated after this step, the calculation proceeds to the next step.

Step III. The Risch algorithm is used. This algorithm determines whether the integral can be evaluated in terms of elementary functions; if this can be done, it is done.

The Risch algorithm is based on a study of the structure of the indefinite integral which dates back to the early part of the nineteenth century. Analyzing integrals of algebraic functions [the function $y(x)$ is "algebraic" if it satisfies the equation $P(x,y)=0$, where P is some polynomial with integer coefficients], Laplace suggested that an integral contains only those algebraic functions which are present in the integrand. This suggestion was subsequently proved by Abel. Then Liouville studied the form of an integral of various combinations of elementary functions in a series of papers in the 1830s and 1840s.

As a result of this work, Liouville proposed that if a function $f(x)$ belongs to some field F of elementary functions, and if the integral of the function $f(x)$ is again an elementary function, then this integral can be represented by a finite sum of the type

$$\int f(x)\,dx = V_0(x) + \sum_{i=1}^{k} C_i \ln V_i(x), \tag{4.1}$$

where the functions $V_0(x)$ and $V_9(x)$ belong to the same field F as the integrand, and the C_i are constants.

This proposal, known now as the Liouville theorem, was proved rigorously by Risch[123] in 1969 and became the basis of a fundamental algorithm which Risch developed for integrating elementary functions. We note that the logarithmic terms in Eq. (4.1) appear only when the integrand has a fractional part, and these terms are governed by the structure of the denominator of the integrand. That this is true is easily seen in the example of integrals of rational functions (forming a field), for which the functions $V_i(x)$ are the same as the factors of the denominator which are linear in x.

We will not go into the mathematical details of the Risch logarithm here (the reader is referred to Refs. 123, 124, and 127), and we will discuss a slightly simplified scheme of a recent modification of this algorithm,[130] which substantially simplifies the evaluation of integrals of transcendental functions. It was this version of the Risch algorithm which Harrington used in his REDUCE-2 program.[129] We recall that implementing the Risch algorithm in the case of algebraic functions is a problem in its own right, which we will not discuss here.

The Risch algorithm is implemented through the following procedure.

I. The integrand is converted to the form p/q, where the functions p and q are polynomials in structures such as $x, e^x, \ln x, \ldots$, which are independent with respect to field operations (i.e., summation, multiplication, and division). To simplify the notation, we denote these structures by $x_1, x_2, x_3, \ldots x_N$. Here N is the number of structures. We will need to distinguish between the variables x_i which are of logarithmic and exponential types.

II. In accordance with the Liouville theorem, (4.1), we write the integral as

$$\int \frac{p}{q}\,dx = \frac{u(x_1, x_2, \ldots, x_N)}{\hat{q}} + \sum_i C_i \ln q_i; \tag{4.2}$$

where u is an unknown polynomial, the C_i are constants, and the polynomials q_i are the irreducible factors of the polynomial q:

$$q = \prod_i q_i^{n_i}.$$

The polynomial $\hat{q}$ is expressed in terms of the factors q_i by

$$\hat{q} = \prod_i q_i^{n_i - 1},$$

except in the case in which any of the factors q_i is equal to a variable x which is of the exponential type. In this case, the exponent on the factor q_i remains equal to n_i (it is not reduced by one); the corresponding logarithmic term does not appear in Eq. (4.2).

III. Differentiating (4.2), we find a first-order linear differential equation in the unknown polynomial u.

IV. Substituting into the differential equation found in step III the polynomial u, written as a sum of the type

$$u(x_1, \ldots, x_N) = \sum_{j_1, \ldots, j_N} u_{j_1 \ldots j_N} x_1^{j_1} \cdots x_N^{j_N}, \qquad (4.3)$$

we find recurrence relations for the coefficients $u_{j_1 \ldots j_N}$.

V. Using these recurrence relations, we can determine which term in the sum in (4.3) has the highest power, $j_1 + j_2 + \cdots + j_N$. Then on the right side of Eq. (4.2) we separate the corresponding term from the original integral, finding new recurrence relations for the coefficients of the remaining part of the polynomial u.

VI. We repeat the calculations in steps IV and V, successively reducing the power of the unknown part of the sum in (4.3) and making use of the arbitrariness in the choice of the parameters C_i, if necessary. Ultimately, we either complete the calculation of the polynomial u and the parameters C_i in Eq. (4.2), i.e., we evaluate the original integral, or in some step we arrive at a contradiction of the results of the preceding calculations. This situation would mean that the original integral cannot be written in the form in (4.2), so that it cannot be expressed in terms of elementary functions, by virtue of the Liouville theorem. Even in this case, however, the calculations are useful, since they make it possible to distinguish the "nonelementary" part of the original integral.

To illustrate the operation of the Risch algorithm for a specific example, we will attempt to evaluate the integral

$$I = \int x^2 e^{x^2} \, dx$$

by the algorithm described above.

I. Clearly, the independent structures in the integrand are x and e^{x^2}. Adopting the notation $e^{x^2} = y$, we can write the integrand in the form $x^2 y$.

II. Since the integrand is a polynomial (does not contain a denominator), Eq. (4.2) gives us

$$I = u(x, y),$$

where u is an unkown polynomial.

III. Differentiating, we find

$$u'(x, y) = x^2 y.$$

IV. Assuming $u(x, y) = \sum u_{ij} x^i y^j$, and evaluating the derivative

$$u'(x, y) = \sum_{ij} u_{ij} (i x^{i-1} y^j + 2j x^{i+1} y^j) = \sum_{ij} x^i y^j [(i+1) u_{i+1, j} + 2j u_{i-1, j}],$$

we find the recurrence relations

$$(i + 1) u_{i+1, j} + 2j u_{i-1, j} = \delta_{i, 2} \delta_j,$$

where δ_{ij} is the Kronecker delta.

V. The single recurrence relation with a nonvanishing right side is

$$3u_{3,1} + 2u_{1,1} = 1.$$

It follows that the coefficient of the polynomial $u(x, y)$ for the maximum power x, y must be $u_{1,1} = 1/2$. Otherwise, the recurrence relations would have led to nonvanishing coefficients $u_{2k+1,1}$ for all $k \geq 1$, in contradiction of the finite nature of the sum in (4.3), which represents the polynomial.

There is accordingly a term $xy/2$ on the right side of Eq. (4.2). Separating the corresponding term from the original integral, we find

$$\int x^2 e^{x^2} \, dx = \frac{1}{2} x e^{x^2} - \int \frac{1}{2} e^{x^2} \, dx.$$

VI. We repeat the calculations in steps IV and V for the remaining integral,

$$\int \frac{1}{2} e^{x^2} \, dx.$$

As a result we find

$$u_{1,1} + 2u_{-1,1} = -\frac{1}{2}.$$

We must equate the coefficient $u_{-1,1}$ to zero, since $u(x, y)$ is a polynomial. However, the equation $u_{1,1} = -1/2$, which overdetermines the result found previously, cannot be ruled out, since the polynomial corresponding to the remaining integral must have a degree lower than that of the original polynomial.

As a result we conclude that the original integral cannot be evaluated in terms of elementary functions, and the representation found for this integral in step V should be accepted as the final result.

These powerful tools for evaluating indefinite integrals are now embodied in the MACSYMA and SCRATCHPAD systems and in the new version of REDUCE-2, i.e., in the universal analytic programming systems. This situation did not, of course, arise by chance. These tools, primarily the Risch algorithm, require the very subtlest methods of analytic programming.

The next important step is to develop a Risch algorithm for algebraic functions that uses the concepts of modern algebraic geometry. This problem is currently the subject of much study,[130] and it will apparently be solved in the near future. Another important direction in the development of the Risch algorithm is a generalization to special functions.[127,131]

b) Solution of differential equations

The use of computers for analytic solution of differential equations has been the subject of a long list of papers (see, for example, the bibliographies in Refs. 122, 132–134, and 136). The MACSYMA system[24] incorporates a special ODE-2 unit,[137] which can solve a broad range of first-order and second-order ordinary differential equations.

As an example, we will examine the method for solving a first-order equation,

$$f(x, y) y' + g(x, y) = 0, \qquad (4.4)$$

as used in the EULE program, written by Schmidt[134] in the PL/1 algorithmic language.[4] No general method for solving Eq. (4.4) is currently available. The usual pro-

cedure is to change variables or to multiply Eq. (4.4) by some expression, in the hope that this equation can be reduced to one with separable variables, a linear equation, or an equation in total differentials.

The EULE program is based on a heuristic approach of this type; the solution of a differential equation of the type in (4.4) is sought by the following procedure.

I. An examination is made to see whether this equation is

A) An equation with separable variables, a homogeneous equation, or an equation with coefficients which are linear in x and y;

B) the linear equation $f_0(x)y'+f_1(x)y+f_2(x)=0$; the Bernoulli equation $f_0(x)y'+f_1(x)y+f_2(x)y^c=0$, $c=\text{const}\neq 1$, the Riccati equation; $f_0(x)y'+f_1(x)y^2+f_2(x)y+f_3(x)=0$; the nearly linear equation $f_0(x)h'(y)y'+f_1(x)h(y)+f_2(x)=0$; an equation of the type

$$y' = ay^n + bx^{n/(1-n)}$$

or

$$y' = \frac{f^{1-n}(x)\,g'(x)}{(ag(x)+b)^n}\,y^n + \frac{f'(x)}{f(x)}\,y + f(x)\,g'(x).$$

If the equation is the Riccati equation, the program attempts to find a particular solution $y_0(x)$ with which the original equation can be reduced to a linear equation by the means of the substitution $\bar{y}=1/(y-y_0)$;

C) an equation in total differentials. To determine whether this is the case, an attempt is made to find an integral factor of the type $x^m y^n$, or a function of x alone, or a function of y alone.

II. The interchange $x \rightleftarrows y$ is used, and test I is repeated.

III. A change of variables is selected on the basis of the particular expressions in the equation. For example, if there are terms of the type $[\varphi(x,y)]^c$, where c is a constant, the replacement $\bar{y}=\varphi(x,y)$, $\bar{x}=x$ is used. If, on the other hand, there are terms of the type $F[\psi(y)]$, then the replacement $\bar{y}=F[\psi(y)]$, $\bar{x}=x$ is used. If F is a trigonometric function, the following substitution is used:

$$\bar{y}=\cos(\psi(y)),\quad \bar{y}=\operatorname{tg}(\psi(y)),\quad \bar{y}=\operatorname{tg}\left(\frac{\psi(y)}{2}\right).$$

The analysis of step I is repeated after each substitution.

IV. The replacements $\bar{y}=yx$, $\bar{y}=y/x$ are made, and test I is applied again after each replacement.

V. An attempt is made to choose integrating factors which are functions of the type

$$x^i \pm y^j \quad \text{or} \quad x^i y^j \qquad (i, j = 1, 2, 3).$$

The method for solving the differential equation in (4.4) which is used in the EULE program operates through an analysis of the equations covered in the familiar handbook by Kamke[138] or the book by Murphy.[139]

The EULE program was tested by using 1245 equations from four collections of differential equations.[138-141] The results are shown in Table V. An equation is judged solved if the program finds a solution method in accordance with the procedure outlined above.

TABLE V. Results of a test of the EULE program.

Collection of equations	Number of equations given	Solved	Collection of equations	Number of equations given	Solved
Kamke[138]	333	90%	Ince[140]	121	100%
Murphy[139]	715	95%	Spiegel[141]	76	100%

Note. The collections by Ince and Spiegel are actually problem sets for second- and third-year university students.

Among the differential equations in the books by Kamke and Murphy, 171 and 505, respectively, can be solved by elementary methods (separation of variables, etc.). The EULE program also used elementary methods to solve the equations in 100% and 99.4% of the cases, respectively.

Eighteen and 23 of the Riccati differential equations in Kamke's and Murphy's books, respectively, are solved by seeking a particular solution. The EULE program solved these equations in the same way.

Twenty-five and 71 of Kamke's and Murphy's differential equations, respectively, are solved by choosing integrating factors. The program used the same method in 92% and 90%, respectively, of the cases.

Frequently, the equation presented to the program was solved by a method which had not been used previously for that particular equation. In some cases, the solution method used by the program turned out to be more elegant and more efficient than the method used in the literature. For example, to solve the Riccati equation

$$xy' \ln x - y^2 \ln x - (2\ln^2 x + 1)\,y - \ln^3 x = 0 \qquad (4.5)$$

Kamke uses the substitution $y=u(s)$, $s=\ln x$ and then the substitution $y=-v'/v$. The result is to convert Eq. (4.5) into a second-order equation. The program, on the other hand, found the particular solution $y=-\ln x$ for this equation, and then it was a trivial matter to find the general solution.

Where the EULE program was capable of solving the given differential equation, the solution generally required no more than 15 sec on an IBM 370/168 (Table VI).

Those differential equations which the program was not able to solve can be put in the following groups:

1) those which can be solved by a change of variables which the program was not able to find;

2) those which can be solved by a series of successive substitutions;

TABLE VI. Typical examples of the computer time required to solve the differential equations from Kamke's collection[138] by the EULE program.

Equation no.	1.348	1.164	1.33	1.313	1.293	1.120
Time, sec	0.5	2.5	3.2	27.0	1.0	1.5

3) those which can be solved by some method which is not programmed in the EULE.

c) Analysis of mathematical expressions

Both mathematicians and physicists are frequently forced to analyze some mathematical expression qualitatively. The problem is to find answers to several questions such as the following.

Is the given expression real, bounded, positive, continuous, monotonic, differentiable?

Are there any singularities, zeros, or local extrema in the expression, and if so, where are they? What is their order?

Is there a simple asymptotic representation for the case in which some of the arguments approach zero or infinity?

When the mathematical expression is complicated, it becomes a difficult, time-consuming problem to answer these questions. Stoutemyer[142] has developed a MACSYMA program[24] especially for analyzing mathematical expressions. This program determines the zeros and singularities of a given expression, the value of the expression as one of the variables approaches a given limit, the extrema, the extreme values of the expression, and its upper and lower bounds. Furthermore, the Stoutemyer program yields information about whether the given expression is decreasing, increasing, or constant on a specified interval; whether it is convex or concave; and whether it is even or odd. The program also determines whether the expression is periodic in some variable; if it is, the period is found.

Although this program realizes by no means all the possibilities of the MACSYMA system for qualitative and quantitative analysis of mathematical expressions, it undoubtedly takes a big step toward simplifying and speeding up this effort.

5. CONCLUSION

The examples which we have discussed in this review of course do not come close to exhausting the range of applications of analytic programming systems in physics and mathematics, which have been the subject of more than 500 publications. Some of the papers which we have not discussed here are covered in the reviews in Refs. 40, 43, 59, 61–65, 92, and 127. Furthermore, we have not discussed applications in such fields of physics as geophysics,[143] crystal physics,[144] atmospheric physics,[145] and cosmology.[146].

In mathematics, our coverage has been even less complete. We might note in particular that there are applications in solving integral,[147] difference,[148] and functional[149] equations; in the theory of commutative rings[150]; and many aspects of group theory.[135] A good idea about other fields of application can be obtained from Refs. 39, 88, and 135.

There can be no doubt that the number of problems which can be solved by analytic programming systems will continue to increase. The process will be aided considerably by a more widespread adoption of the systems, which in turn depends on their adaptability to different computers. In contrast with numerical programs, the programs and systems for computer analytics are closely tied to the internal structure of the particular computer. Serious difficulties thus arise in attempts to switch from computers of one type to another, and the way in which the difficulties are resolved differs from case to case (see, for example, Fateman's discussion[151] of the problems involved in switching the MACSYMA system from a DEC PDP-10 to a CDC 6600/7600).

The most adaptable of the systems in Table I is undoubtedly REDUCE-2, which is the one which has been adopted most widely.[152] Computer analytics will not find really widespread use, of course, until a fundamental solution is found for the computer-memory problem which results from the increase in the size of the intermediate calculations. This fundamental solution would make it possible to use powerful analytic programming systems on small computers.

Another trend which is expanding the range of applications of analytic programming systems is their use in combined numerical–analytic calculations. This approach is particularly valuable for those problems which either cannot be solved analytically or require an unacceptable amount of computer time for analytic solution, but for which ordinary numerical calculations are not adequate because of the unacceptably large calculation errors. Some examples of these problems are discussed in Section 3c (the contribution of nonplanar seventh-order vertex diagrams to the anomalous magnetic moment of the electron) and Section 3d (analysis of the plasma stability in tokamaks).

Each analytic programming system, of course, also has provision for dealing with purely numerical calculations; nearly all the systems can operate with rational numbers (see, for example, Table II); and nearly all can perform arithmetic operations on these numbers without any errors (the exact arithmetic of integers and rational numbers). However, the numerical calculations carried out by analytic programming systems are much slower than those carried out by ordinary numerical systems, such as FORTRAN and ALGOL. This circumstance is a serious problem in the interaction between analytic and numerical systems.

In summary, analytic programming systems must be developed in two directions in order to make effective use of computers for combined numerical–analytical

```
PRINT NLIST
PRINT NSTAT
F A
Z DET3=DS(J,1,3,(DS(K,1,3,(DS(L,1,3,(A(1,J)*A(2,K)*A(3,L)
 *DP(J,K,L)))))))
*END
DET3=
  +A(1,1)*A(2,2)*A(3,3)
  -A(1,1)*A(2,3)*A(3,2)
  -A(1,2)*A(2,1)*A(3,3)
  +A(1,2)*A(2,3)*A(3,1)
  +A(1,3)*A(2,1)*A(3,2)
  -A(1,3)*A(2,2)*A(3,1)+0.
```

FIG. 5. SCHOONSCHIP program for calculating a third-order determinant.

```
ARRAY P(3);
FOR I:=1:3 DO BEGIN P(I):=(X**2-1)**I;
FOR J:=1:I DO P(I):=DF(P(I),X)/(2*J);
WRITE P(,I,) = ,P(I) END;

P(1)=X

                2
P(2)=(3*X  -1)/2

                    2
P(3)=(X*(5*X  -3))/2

END
```

FIG. 6. REDUCE-2 program for generating Legendre polynomials from the Rodrigues formula.

calculations.

I. The analytic programming systems must be modified to improve their interaction with numerical programming systems. These improvements will not only speed up many calculations but will also open up the possibility of using the extensive libraries of standard numerical programs.

II. The methods used for the numerical calculations in the analytic programming systems themselves must be improved, in order to increase the speed in an accurate solution of any problem. As an important step in this direction we might cite the paper by Sasaki,[153] who has developed a new packet for exact arithmetic in REDUCE-2.

Clearly, a harmonious blend of the capabilities of modern computers in terms of numerical and analytic calculations will make these machines capable of solving an extremely broad range of problems in various fields of knowledge.

6. SOME SIMPLE PROGRAMS IN SCHOONSCHIP AND REDUCE-2

To illustrate the practical use of analytic programming systems, we will consider two simple examples.

a) First, we will examine the use of SCHOONSCHIP to calculate a third-order determinant.

For the calculations we use the formula

$$\det \| a_{ij} \| = \sum_{j=1}^{3} \sum_{k=1}^{3} \sum_{l=1}^{3} a_{1j} a_{2k} a_{3l} \varepsilon_{jkl}, \tag{6.1}$$

where ε_{jkl} is the Levi–Civitá symbol (the completely anti-symmetric tensor; $\varepsilon_{123} = 1$). In the SCHOONSCHIP system, this tensor corresponds to the incorporated function $DP(J, K, L)$. To carry out the summation, the system function DS is used. The program which implements Eq. (6.1) in SCHOONSCHIP and the machine printprintout of the result of the calculation are shown in Fig. 5.

b) The second application which we will consider is the generation of Legendre polynomials by means of REDUCE-2.

For the calculations we use the Rodrigues formula

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} [(x^2 - 1)^n]. \tag{6.2}$$

The program for generating the polynomials $P_n(x)$ with $n = 1-3$ from Eq. (6.2), which uses the differentiation operator $DF(Q, x) \equiv dQ/dx$, which is embodied in the REDUCE-2 system, is shown along with the calculated results in Fig. 6.

We wish to thank M. G. Meshcheryakov, N. N. Govorun, V. A. Brumberg, D. P. Kostomarov, V. A. Meshcheryakov, É. A. Perel'shteĭn, I. V. Potgosin, V. A. Rostovtsev, R. N. Fedorova, and V. P. Shirikov for consultations regarding the SYMBAL system. We also thank É. D. Krupnikov for assistance in compiling the bibliography.

[1] P. Morrison and E. Morrison (editors), Charles Babbage and His Calculating Engines, Dover, New York, 1961.
[2] H. G. Kahrimanian, Analytical Differentiation by a Digital Computer, MA Thesis, Temple Univ., Philadelphia, 1953; J. Nolan, Analytical Differentiation on a Digital Computer, MA Thesis, MIT Cambridge, Mass., 1953.
[3] D. E. Knuth, The Art of Computer Programming, Vol. 2, Addison–Wesley, Reading, Mass., 1969 (Russ. Transl., Mir, Moscow, 1977).
[4] N. A. Krinitskiĭ, G. A. Mironov, and G. D. Frolov, Programmirovanie i algoritmicheskie yazyki (Programming and Algorithmic Languages), Nauka, Moscow, 1979.
[5] W. D. Maurer, Programmer's Introduction to LISP, American Elsevier, New York, 1972 (Russ. Transl., Mir, M., 1976); S. S. Lavrov and G. S. Silagadze, Avtomaticheskaya obrabotka dannykh: Yazyk LISP i ego realizatsiya (Automatic Data Processing: The LISP Language and Its Implementation), Nauka, Moscow, 1978.
[6] J. McCarthy, Comm. ACM 3, 184 (1960).
[7] J. Strubbe, Comp. Phys. Comm. 8, 1 (1974).
[8] G. E. Collins, in: Proc. of the Second Symposium on the Symbolic and Algebraic Manipulation, ACM, New York, 1971, p. 144.
[9] D. Lurie, in: Computing as a Language of Physics, IAEA, Vienna, 1972, p. 529.
[10] A. C. Hearn, REDUCE User's Manual, Second Edition, Univ. of Utah, 1973.
[11] A. C. Hearn, in: Proc. of the Third Intern. Colloquium on Advanced Computing Methods in Theoretical Physics, Vol. 1, C.N.R.S., Marseilles, 1973, p. A-V-1.
[12] R. C. Perisho, ASHMEDAI User's Guide, U.S.A.E.C. Rept. No. COO-3066-44, 1975.
[13] R. A. D'Inverno, The ALAM Programmer's Manual, King's College, London, 1969.
[14] R. A. D'Inverno and R. A. Russel-Clark, The CLAM Programmer's Manual. Part 1, King's College, London, 1971; Part 2, Univ. of Cambridge Computer Laboratory, Cambridge, 1972.
[15] I. Frick, SHEEP User's Guide. USIP Report 77-15, Univ. of Stockholm, Stockholm, 1977.
[16] A. Rom, Celest. Mech. 1, 301 (1969).
[17] V. A. Brumberg and L. A. Isakovich, in: Algoritmy nebesnoĭ mekhaniki (Algorithms for Celestial Mechanics), No. 1, ITA Akad. Nauk SSSR, Leningrad, 1974.
[18] J. P. Fitch, CAMAL User's Manual, Univ. of Cambridge Computer Laboratory, Cambridge, 1975.
[19] E. A. Araĭs and G. V. Sibiryakov, AVTO-ANALITIK (The AVTO-ANALITIK Analytic Programming Language), Izd. Novosibirsk. Univ., Novosibirsk, 1973.
[20] N. A. Trufyn, Guide to FORMAC, Univ. of Toronto, Toronto, 1970.
[21] W. S. Brown, ALTRAN User's Manual, Third Edition, Bell Lab, 1973.
[22] M. E. Engeli, SYMBAL User's Manual, Univ. of Texas, 1969.
[23] V. M. Glushkov, T. A. Grinchenko, A. A. Dorodnitsyna, A. M. Drakh, Yu. V. Kapitonova, V. P. Klimenko, L. N. Kres, A. A. Letichevskiĭ, S. B. Pogrebinskiĭ, O. N. Savchak, A. A. Stognii, Yu. S. Fishman, and N. P. Tsaryuk, Kiber-

netika 5, 114 (1978).
[24]R. Bogen, J. Golden, M. Genesereth, and A. Doohovskoy, MACSYMA Reference Manual, Version Nine, MIT Mathlab Group, Lab. for Computer Science, 1977.
[25]J. H. Griesmer, R. D. Jenks, and D. Y. Y. Yun, SCRATCHPAD User's Manual. IBM Research Rept. RA-70, 1975.
[26]J. Korpela, Helsinki Univ. of Technology Rept. No. 27, 1976.
[27]L. V. Kantorovich and L. T. Petrova, in: Trudy tret'ego Vsesoyuznogo matematicheskogo s"ezda (Proceedings of the Third All-Union Mathematics Congress), Vol. 2, Nauka, Moscow, 1956, p. 151.
[28]L. V. Kantorovich, Izv. Akad. Nauk ArmSSR 10, No. 2 (1957).
[29]L. V. Kantorovich, Dokl. Akad. Nauk SSSR 113, 738 (1957).
[30]L. T. Petrova, Izv. Vyssh. Uchebn. Zaved., Matematika 5, 95 (1958).
[31]V. A. Bulavskiĭ, Izv. Vyssh. Uchebn. Zaved., Matematika 5, 5 (1958).
[32]T. N. Smirnova, Provedenie na ÉVM tipa M-20 polinomial'nykh vykladok s pomoshch'yu prorabov (Polynomial Calculations on an M-20 Computer by a "Foreman" Method), Nauka, Leningr. Otd-nie, Leningrad, 1967.
[33]V. A. Shurygin and N. N. Yanenko, in: Problemy kibernetiki (Problems of Cybernetics), No. 6, Fizmatgiz, Moscow, 1961.
[34]I. P. Aksel'rod and L. F. Belous, Vkhodnoĭ yazyk sistemy avtomaticheskogo programmirovaniya SIRIUS (Input Language of the SIRIUS Automatic Programming System), Izd. Khar'k. Univ., Khar'kov, 1969.
[35]M. M. Bezhanova, V. L. Katkov, and I. V. Pottosin, in Vychislitel'naya matematika i vychislitel'naya tekhnika (Computational Mathematics and Techniques), No. III, FTINT Akad. Nauk UkrSSR, Khar'kov, 1972, p. 18; N. I. Kostyukova, p. 38.
[36]V. S. Synakh, Preprint, Computation Center, Siberian Branch, Academy of Sciences of the USSR, Novosibirsk, 1965.
[37]V. S. Verebryusov, Preprint ITÉF-54, Institute of Theoretical and Experimental Physics, Moscow, 1974.
[38]Vychislitel'naya matematika i vychislitel'naya tekhnika (Computational Mathematics and Techniques), No. III, FTINT Akad. Nauk UkrSSR, Khar'kov, 1972.
[39]Proceedings of the 1977 MACSYMA User's Conference, Berkeley, California, 27–29 July, NASA Scientific and Technical Information Office, Washington, 1977.
[40]D. Barton and J. P. Fitch, Rep. Prog. Phys. 35, 235 (1972).
[41]M. S. Davis, Astron. J. 63, 462 (1958).
[42]P. Herget and P. Musen, Astron. J. 64, 11 (1959).
[43]W. H. Jeffrys, Comm. ACM 14, 538 (1971).
[44]C. Delaunay, Theorie du Mouvement de la Lune: Extraits des Mem. Acad. Sci., Mallet–Bachelier, Paris, 1860.
[45]A. Deprit, J. Henrard, and A. Rom, Science 168, 1569 (1970).
[46]V. A. Brumberg and L. A. Isakovich, in: Algoritmy nebesnoĭ mekhaniki (Algorithms for Celestial Mechanics), No. 4, ITA Akad. Nauk SSSR, Leningrad, 1975.
[47]A. V. Vasil'eva, in: Algoritmy nebesnoĭ mekhaniki (Algorithms for Celestial Mechanics), No. 7, ITA Akad. Nauk SSSR, Leningrad, 1975.
[48]L. S. Evdokimova, Algoritmy nebesnoĭ mekhaniki (Algorithms for Celestial Mechanics), No. 15, ITA Akad. Nauk SSSR, Leningrad, 1977.
[49]J. D. Anderson and E. L. Lau, Proceedings of the 1977 MACSYMA User's Conference, Berkeley, California, 27–29 July, NASA Scientific and Technical Information Office, Washington, 1977, p. 395.
[50]J. G. Fletcher, R. W. Clemens, R. A. Matzner, K. S. Thorne, and B. A. Zimmerman, Astrophys. J. Lett. 148, 91 (1967).
[51]H. Bondi, M. Van Der Burg, and A. Metzner, Proc. R. Soc. London, Ser. A 269, 21 (1962).
[52]G. H. Harrison, J. Comp. Phys. 4, 594 (1969).
[53]R. A. d'Inverno and R. A. Russell-Clark, J. Math. Phys. 12, 1258 (1971).
[54]J. P. Fitch, Univ. of Cambridge Report, Cambridge, 1971.
[55]A. Z. Petrov, Prostranstva Einshteĭna (Einstein Spaces,) Fizmatgiz, Moscow, 1961 (Pergamon, New York, 1969).
[56]G. W. Gibbons and R. A. Russell–Clark, Phys. Rev. Lett. 30, 398 (1973).
[57]I. Cohen, J. Comp. Phys. 22, 396 (1976).
[58]R. A. d'Inverno, Gen. Relativ. Gravit. 6, 567 (1975).
[59]H. I. Cohen, Ö. Leringe, and Y. Sundblad, Gen. Relativ. Gravit. 7, 269 (1976).
[60]I. E. Zhidkova, I. P. Nedyalkov, and V. A. Rostoftsev, Preprint R2-11589, Joint Institute for Nuclear Research, Dubna, 1978.
[61]A. C. Hearn, in: Computing as a Language of Physics, IAEA, Vienna, 1972, p. 567.
[62]J. A. Campbell, Acta Phys. Austriaca Suppl. 13, 595 (1974).
[63]A. C. Hearn, in: Proc. of the 1976 CERN School of Computing. La Grande Motte, France, 12–15 September, CERN 76-24, Geneva, 1976, p. 201.
[64]A. C. Hearn, in: Trudy Mezhdunarodnogo soveshchaniya po programmirovaniyu i matematicheskim metodam resheniya fizicheskikh zadach (Proceedings of the International Conference on Programming and Mathematical Methods for Solving Physical Problems), 20–23 September 1977, Dubna, OIYaI, Dubna, D10.11-11264, 1978, p. 96.
[65]V. P. Gerdt, in: Trudy Mezhdunarodnogo soveshchaniya po programmirovaniyu i matematicheskim metodam resheniya fizicheskikh zadach (Proceedings of the International Conference on Programming and Mathematical Methods for Solving Physical Problems), 20–23 September 1977, Dubna, OIYaI, Dubna, D10.11-11264, 1978, p. 166.
[66]N. N. Bogolyubov and D. V. Shirkov, Vvedenie v teoriyu kvantovannykh poleĭ (Introduction to Quantum Field Theory), Nauka, Moscow, 1976.
[67]A. I. Akhiezer and V. B. Berestetskiĭ, Kvantovaya elektrodinamika (Quantum Electrodynamics), Nauka, Moscow, 1969.
[68]J. D. Bjorken and S. D. Drell, Relativistic Quantum Mechanics, Vol. 2, McGraw–Hill, New York, 1964 (Russ. Transl., Nauka, Moscow, 1978).
[69]J. A. Campbell and A. C. Hearn, J. Comp. Phys. 5, 280 (1970).
[70]J. Calmet and M. Perrottet, J. Comp. Phys. 7, 191 (1971).
[71]M. Perrottet, in: Computing as a Language of Physics, IAEA, Vienna, 1972, p. 555.
[72]T. Sasaki, J. Comp. Phys. 22, 189 (1976).
[73]J. S. R. Chisholm, Nuovo Cimento 30, 426 (1963).
[74]J. Kahane, J. Math. Phys. 9, 1732 (1968).
[75]N. N. Bogolyubov and O. S. Parasyuk, Izv. Akad. Nauk SSSR, Ser. Matem. 20, 585 (1956); O. S. Parasyuk, Izv. Akad. Nauk SSSR, Ser. Matem. 20, 843 (1956); N. N. Bogoliubow (Bogolyubov) and O. S. Parasiuk, Acta Math. 97, 227 (1957).
[76]V. I. Kucheryavyĭ, Preprint ITF-76-132R, Institute of Theoretical Physics, Kiev, 1976.
[77]J. Calmet, SIGSAM Bulletin (ACM, N.Y.), No. 31, p. 74 (1974).
[78]O. V. Tarasov, Preprint E2-11573, JINR, Dubna, 1978.
[79]B. E. Lautrup, in: Proc. of the Second Colloquium on Advanced Computing Methods in Theoretical Physics, Vol. 1, C.N.R.S., Marseilles, 1971, pp. 1–58.
[80]M. J. Levine and R. Roskies, Phys. Rev. D9, 421 (1974); M. J. Levine, R. C. Prisho, and R. Roskies, Phys. Rev. D13, 997 (1976).
[81]C. M. Bender, R. W. Keener, and R. E. Zippel, Phys. Rev. D15, 1572 (1977).
[82]R. Barbieri, J. A. Mignaco, and E. Remiddi, Nuovo Cimento A11, 824 (1972).
[83]R. Barbiere and E. Remiddi, Nucl. Phys. B90, 233 (1975); R. Barbieri, M. Caffo, and E. Remiddi, Phys. Lett. B57, 460 (1975).
[84]D. Maison and A. Petermann, Comp. Phys. Comm. 7, 121 (1974).
[85]J. A. Fox and A. C. Hearn, J. Comp. Phys. 14, 301 (1974).

[86]D. Yu. Bardin, O. M. Fedorenko, and N. M. Shumeĭko, Preprint R2-10114, Joint Institute for Nuclear Research, Dubna, 1976.
[87]V. Linke, E. Tränkle, and I. Bender, Nuovo Cimento **A42**, 281 (1977); in: Proceedings of the Fourth Intern. Colloquium on Advanced Computing Methods in Theoretical Physics, Saint-Maximum, France, 1977, p. 133.
[88]Proceedings of the Fourth Intern. Colloquium on Advanced Computing Methods in Theoretical Physics, Saint-Maximum, France, 1977.
[89]J. Schwinger, Phys. Rev. **73**, 416 (1948); **76**, 790 (1949).
[90]A. Petermann, Helv. Phys. Acta **30**, 407 (1957).
[91]C. M. Sommerfeld, Ann. Phys. (NY) **5**, 26 (1958).
[92]J. Calmet, in: Proc. of the Third Intern. Colloquium on Advanced Computing Methods in Theoretical Physics, Vol. II, C.N.R.S., Marseilles, 1973, p. C-I-1.
[93]M. Levine, E. Remiddi, and R. Roskies, in: Proceedings of the Fourth Intern. Colloquium on Advanced Computing Methods in Theoretical Physics, Saint-Maximum, France, p. 178.
[94]T. Konoshita, in: Proc. of the Nineteenth Intern. Conference on High Energy Physics, 23–30 August 1978, Tokyo, Physical Society of Japan, 1979, p. 571.
[95]M. J. Levine and J. Wright, Phys.Rev. **D8**, 3171 (1973).
[96]R. S. Van Dyck, Jr., P. B. Schwinberg, and H. G. Dehmelt, Phys. Rev. Lett. **38**, 310 (1977).
[97]T. Kinoshita, Preprint CLNS-390, Cornell University, N.Y., 1978.
[98]A. A. Valdimirov and O. V. Tarasov, Yad. Fiz. **25**, 1104 (1977) [Sov. J. Nucl. Phys. **25**, 858 (1977)].
[99]E. Egorian and O. V. Tarasov, Preprint E2-11757, Joint Institute for Nuclear Research, Dubna, 1978.
[100]A. Billoire, R. Lacaze, A. Morel, and H. Navalet, Phys. Lett. **B78**, 149 (1978).
[101]G. Kuppters, D. Pfirsch, and H. Tasso, in: Plasma Physics and Controlled Nuclear Fusion Research, Vol. 2, IAEA, Vienna, 1971, p. 529.
[102]W. Kerner and J. Steuerwald, Comp. Phys. Comm. **9**, 337 (1975).
[103]W. Kerner, Max-Planck-Institut für Plasmaphysik Report, Munich, 1978.
[104]B. Rosen, J. Comp. Phys. **15**, 98 (1974); **20**, 22 (1976).
[105]C. F. F. Karney, in: Proceedings of the 1977 MACSYMA User's Conference, Berkeley, California, 27–29 July, NASA Scientific and Technical Information Office, Washington, 1977, p. 377.
[106]J. L. Kulp, in: Proceedings of the 1977 MACSYMA User's Conference, Berkeley, California, 27–29 July, NASA Scientific and Technical Information Office, Washington, 1977.
[107]R. W. Atherton and G. M. Homsy, J. Comp. Phys. **13**, 45 (1973).
[108]A. Acrivos and D. Barthès-Biesel, J. Comp. Phys. **12**, 403 (1973).
[109]S. K. Chow, A. H. How, and L. Landweber, Westinghouse Research Laboratory Rept. 75-160-LANCH-P1, Pittsburg, 1975.
[110]S. K. Chow, A. H. How, and L. Landweber, J. Hydronautics **10**, 2 (1976).
[111]I. Cohen and F. Bark, Comp. Phys. Comm. **14**, 319 (1978).
[112]F. H. Bark, P. S. Jeijer, and H. I. Cohen, Phys. Fluids **21**, 531 (1978).
[113]P.-O. Nerbrant, Comp. Phys. Comm. **14**, 315 (1978).
[114]R. Benesch, J. Phys. **B4**, 1402 (1971); Phys. Rev. **A6**, 573 (1972).
[115]G. Rudnicki-Bujnowski, Comp. Phys. Comm. **10**, 245 (1975).
[116]A. P. Yutsic and A. A. Bandzaĭtis, Teoriya momenta kolichestva dvisheniya v kvantovoĭ mekhanike (Angular Momentum Theory in Quantum Mechanics), Mokslas, Vil'nyus, 1977.
[117]H. I. Cohen and S. Yu. Slavyanov, J. Comp. Phys. **29**, 289 (1978).
[118]E. U. Condon and G. H. Shortley, Theory of Atomic Spectra, Cambridge Univ. Press, 1951 (Russ. Transl., IL, M., 1949).
[119]C. F. Fischer and D. W. B. Prentice, Comp. Phys. Comm. **6**, 157 (1973).
[120]L. B. Golden, Comp. Phys. Comm. **14**, 255 (1978).
[121]J. R. Slagle, J. ACM **10**, 507 (1963).
[122]J. Moses, Report MAC TR-47: Project MAC, MIT, Cambridge, 1967.
[123]R. Risch, Trans. AMS **139**, 167 (1969).
[124]R. Risch, Bull. AMS **76**, 605 (1970).
[125]R. Tobey, Algorithms for Antidifferentiation of Rational Functions: Ph.D. Thesis, Harvard Univ., Cambridge, 1967.
[126]E. Horowitz, in: Proc. of the Second Symposium on the Symbolic and Algebraic Manipulation, ACM, New York, 1971, p. 441.
[127]J. Moses, Comm. ACM **14**, 548 (1971).
[128]A. C. Hearn, REDUCE Newslett. (Univ. of Utah), No. 4, 12 (1978).
[129]S. J. Harrington, Report UCP-57, University of Utah, 1978.
[130]A. C. Norman and P. M. A. Moore, in: Proceedings of the Fourth Intern. Colloquium on Advanced Computing Methods in Theoretical Physics, Saint-Maximum, France, 1977, p. 99.
[131]B. F. Caviness, Proceedings of the Fourth Intern. Colloquium on Advanced Computing Methods in Theoretical Physics, Saint-Maximum, France, 1977, p. 16.
[132]A. C. Norman, Comp. J. **19**, No. 1, 63 (1975).
[133]H. Tasso and J. Steuerwald, Preprint IPP 6/143, Max-Planck-Institut für Plasmaphysik, 1976.
[134]P. Schmidt, in: Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation, Yorktown Heights, N. Y., 1976, p. 114.
[135]Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation, Yorktown Heights, N. Y., 1976.
[136]E. L. Lafferty, in: Proceedings of the 1977 MACSYMA User's Conference, Berkeley, California, 27–29 July, NASA Scientific and Technical Information Office, Washington, 1977, p. 347.
[137]J. P. Golden, Proceedings of the 1977 MACSYMA User's Conference, Berkeley, California, 27–29 July, NASA Scientific and Technical Information Office, Washington, 1977, p. 1.
[138]E. Kamke, Solutions and Solution Methods for Differential Equations (Russ. Transl., Nauka, M., 1976).
[139]G. M. Murphy, Ordinary Differential Equations, Princeton, N. J., 1960.
[140]E. L. Ince, Die Integration gewöhnlicher Differentialgleichungen, HTB Nr. 67, Bibliogr. Inst., Mannheim, 1956.
[141]M. R. Spiegel, Applied Differential Equations, Prentice Hall, Englewood Cliffs, N. J., 1958.
[142]D. Stoutemyer, in: Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation, Yorktown Heights, N. Y., 1976, p. 97.
[143]J. N. Hanson, J. Geophys. Res. **78**, 3260 (1973).
[144]R. Head, in: Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation, Yorktown Heights, N. Y., 1976, p. 126.
[145]J. P. Boyd, J. Atmos. Sci. **35**, 2236 (1978).
[146]M. D. Birrell, Proc. R. Soc. London, Ser. B **361**, 513 (1978).
[147]D. R. Stoutemyer, ACM Trans. Math. Software **3**, No. 2, p. 128 (1977).
[148]J. Ivie, in: Proceedings of the 1977 MACSYMA User's Conference, Berkeley, California, 27–29 July, NASA Scientific and Technical Information Office, Washington, 1977, p. 317.
[149]V. P. Gerdt, Preprint R4-12064, Joint Institute for Nuclear Research, Dubna, 1978.
[150]D. A. Spear, in: Proceedings of the 1977 MACSYMA User's Conference, Berkeley, California, 27–29 July, NASA Scientific and Technical Information Office, Washington, 1977, p. 369.
[151]R. J. Fateman, SIGSAM Bulletin (ACM N.Y.), No. 47, 8 (1978).
[152]REDUCE Newslett. (Univ. of Utah), No. 3, 20 (1978).
[153]T. Sasaki, Univ. of Utah Report, 1978.

Translated by Dave Parsons