Some applications of computing to number theory

Muir, S. T. E.

SOME APPLICATIONS OF COMPUTING TO NUMBER THEORY

by

S. T. E. MUIR

C O N T E N T S

A B S T R A C T

This thesis describes the application of a computer to certain problems in number theory.

Chapter 1 is a general description of the use of computers in this field. Chapter 2 contains the results of computations relating to certain non - congruence subgroups of the full modular group, while chapter 3 describes the use of variable precision arithmetic, mainly in connection with continued fraction expansions.

The last chapter describes a small library of subroutines, the majority of which are written in Fortran. A computer - printed listing of these subroutines is given at the end of the thesis.

CHAPTER 1.

1.1. The electronic digital computer.

This thesis describes the application of an electronic digital computer to certain areas of number theory. Chapter 4 is a description of the library of subroutines which has been assembled and used, in particular, to obtain the results contained in Chapters 2 and 3. In this chapter we discuss generally the use of computers, with the emphasis on number-theoretical problems.

In what follows, the word "computer" will be taken to be a shortened version of the phrase "electronic digital computer". The specific computer used to derive the results contained in this thesis is briefly described in 1.5.

A computer performs calculations according to a set of instructions, called a programme, the execution of which is called a run (or job in Atlas terminology). A programme may differ from run to run or may be altered by itself within a run. This fact implies a considerable degree of flexibility in the type of work that may be performed. Sometimes it may be more advantageous to sacrifice this flexibility for an increase in speed of calculation. An example is D.H. Lehmer's Delay Line Sieve (DLS-127), for use in certain number-theoretical calculations - see Lehmer (13), where the economics of the machine is also considered.

The execution of programme instructions proceeds at a high speed, relative to hand computation which has the undesirable feature of being both unreliable and tedious. In number theory, in particular, the electronic computer is a powerful tool in the process of empirical discovery.

1.2. Number-theoretical computing.

Broadly speaking, we may discern two general types of programme. On the one hand, there is the programme designed to calculate and spend most of its time adding and multiplying, for example; on the other hand, we have the programme that updates files on magnetic tape and spends a relatively small amount of time calculating. In a sense, of course, all operations performed by a computer may be regarded as calculation of one kind or another; but here we distinguish between arithmetical calculation and, say, character manipulation. Some programmes could perhaps be placed in both categories. In a sorting programme, for example, a large amount of time is spent comparing quantities, which essentially involves subtraction and testing.

Number-theoretical programmes are, by their very nature, placed in the first category. In this field, the emphasis is on economy of computation and, in general, little ingenuity need be expended on magnetic tape operations. Because a computer can not only calculate but can make decisions on the result of a calculation, number-theoretical programmes themselves fall into two categories. Thus "pure" calculation only is needed to obtain the algebraic equation satisfied by $j(\frac{-1+i\sqrt{2347}}{2})$ from the values of the $j(\tau_k)$ of (3.5.3). In a factorisation routine, however, decision is an essential ingredient of the programme. This distinction makes it more difficult to determine, beforehand, the computing time required by the second type of programme.

Inevitably, the number-theorist's programme will be concerned with integer quantities while that of the numerical analyst, for example, will be almost diametrically opposite, being concerned with real quantities (in Fortran terminology, real is floating-point). To the number-theorist, there is a

world of difference between N and N+1, N a large integer. The numerical analyst would be pedantic if he were to quibble over such matters. This is perhaps an exaggerated example but it does underline one of the basic differences between number-theoretical computing and what might be termed "ordinary" computing. Atkin (1) has made a similar observation, on low-probability faults in a computer's hardware and their disastrous effects on integer arithmetic.

A computer programme presupposes a problem, which raises the question: what problem? Since a programme is executed in a precisely determined fashion by a computer, the problem must be formulated precisely. At present, the computer can have no idea of what the programmer is attempting to solve and can normally be relied upon to execute instructions, whether or not these are sensible. From the computer's point of view, any results obtained must be the correct ones, though they may appear ridiculous to the programmer. As regards suitable problems, it is all too easy for number-theoretical calculations to "pile up lists of integers in the manner of a magpie" - to use the phrase of Swinnerton-Dyer (19). The calculation of π to a million decimals may be regarded as an interesting exercise in its own right, but it cannot be regarded as an important addition to number theory. Of course this opinion is a function of how important the calculation seems to be at present.

Having formulated the problem, speed of calculation and size of store need to be considered. The size of the problem has to be altered accordingly. In 1800 Gauss's considerable calculations on binary quadratic forms were a major undertaking; these same calculations now-a-days take a matter of minutes and would be regarded as a small "production" run. As

Swinnerton-Dyer (19) remarks, "a calculation which takes 10^6 operations is trivial, and is worth doing even if its results are useless". This sort of calculation takes Atlas about 2 or 3 seconds. Since there is no reason to suppose that either speed or store size has reached its limit, theoretical or otherwise, the size of a "trivial" job must remain a variable quantity. Estimates of running times can generally be obtained by extrapolation of small test runs.

1.3. Detection of errors.

The construction of a computer programme almost always involves the detection and removal of errors in the programme.

Errors can arise in many different ways. Firstly, there may be errors in the logic of the programme. Only the programmer can correct these errors, since only the programmer knows what the problem is and what the logic should be. It may be an error in a formula or an incorrect jump after a decision. The actual results produced often give a clue as to the position of the error and the programme usually requires only a small change for successful running.

Secondly, there are errors in the language in which the programme has been written. These are detected by the compiler which translates the statements of that language into basic machine code. Simple spelling mistakes, omitted brackets and undefined jumps are typical errors.

Thirdly, errors at the execution stage, such as the attempt to take

the square-root of a negative quantity, are often due to incorrect logic or formulae in the programme.

All these errors are typical in computer programmes under construction. But there are subtler types of errors that are very much harder to detect. The programmer, in exasperation, will blame the computer; but, in the great majority of cases, it is the programmer and not the computer who is at fault. An example is the overwriting of a programme by itself, usually with catastrophic results. Exceeding array bounds is another typical fault and can overwrite other arrays and machine code situated nearby. Some compilers insert code to check for just this sort of error. Incorrect computed go-to statements and jumping to locations containing data rather than instructions are similar errors. On Atlas, for example, instructions and numbers require the same number of binary digits for storage, namely 48 bits or one "word". It is possible for the contents of a word containing a sensible number to be interpreted as a sensible instruction. If this word is executed as a result of an error of the above mentioned type, the results may be puzzling, to say the least. Usually, for a variety of reasons, the instruction is illegal and is trapped by the computer, either by hardware or by the supervisory programme. At worst, by sheer bad luck, it could be an instruction to write to an unprotected magnetic tape.

Finally, there are various execution errors such as entering an infinite loop of instructions; in this case a large amount of computing is used or a large amount of output produced. Selective printing of relevant quantities in the programme is usually enough to enable the precise location of the error to be ascertained.

1.4. Subroutine libraries.

A subroutine (alternatively routine, subprogramme, procedure, function) is the name given to a body of code which, when compiled, can be executed by simply stating its identifying name. This avoids having to insert the whole body of the subroutine in a programme whenever it is required. Usually a subroutine will have arguments associated with it, so that calls of the subroutine with different values of the arguments produce different results, thus giving the programmer a powerful and flexible device.

Often a subroutine calls another subroutine; the linkage of the subroutines may be quite involved, especially in a large system. To minimise the effort of assembling all the relevant subroutines needed in a particular run of a programme, the concept of a subroutine library is helpful. All the subroutines that the programmer is ever likely to use in any run are transferred to a magnetic tape (or disc). This collection, or library, of subroutines can be arranged on the magnetic tape in such a way that, in any particular run, only those subroutines that are actually needed are called down from tape and assembled in store. The advantage of a subroutine library over a large manipulatory system has been stressed by Lehmer (13).

A small library of number-theoretic subroutines is described in Chapter 4. This particular library is by no means complete and can be easily updated as and when new routines are written.

1.5. The Atlas computer.

Inasmuch as the results given in this thesis have been obtained by

a combination of the author and an I.C.L. Atlas 1 computer, a brief description of the latter system may be of some interest.

The store of the computer is arranged in units of 48 binary digits, a "word". There are 8192 words of "fixed" read-only store, 16384 words of working store, 49152 words of ferrite core and 98304 words of magnetic drum store. The access times of these stores are respectively 0.8, 2, 2 and 4 μsecs. per word, where 1 μsec. = 10^{-6} second. There are 16 magnetic tape decks each with a transfer rate of approximately 64000 characters per second (8 characters are held in one word). Information is held on magnetic tape in units of 512 words (one "block"). A magnetic tape can contain approximately 5000 blocks. In addition, there is a non-interchangeable magnetic disc file with a storage capacity of approximately 16½ million words.

At any one time, many different programmes may be in the store; only one is actually being executed. If this job is held up for any reason - for example, a request to write to magnetic tape - the computer starts executing another job and is thus kept as busy as possible. In fact, magnetic tape operations, once initiated, proceed independently. All this "organisational" work is performed by a master programme (the supervisor, in Atlas terminology). Because the computer is constantly swapping from one programme to another, in a manner determined by the supervisor, the real time of execution of any particular programme is usually an irrelevant quantity, as far as the programmer is concerned. Thus a programme may need 10 minutes of central processor time yet take an hour to pass through the machine. All the times quoted in this thesis are central processor times, which are in fact recorded in units of 2048

instructions completed (one "interrupt"). One second of computing is approximately 160 interrupts. Typical times are: 2 μsecs. for floating-point addition, approximately 6 μsecs. for floating-point multiplication, and from 1.6 to 1.8 μsecs. for "organisational" instructions.

The supervisor is written in such a way that, to the programmer, the core store and magnetic drum store appear to be continuous; sections of each are swapped onto the other, when necessary, by means of the supervisor's "drum-learning" programme. This programme attempts to perform these swappings with the best possible efficiency. Only in a few special cases is it necessary to take account of the fact that the computer's store is really on two levels. Thus the execution of EULMUL with a large number of terms results in a prohibitive number of drum transfers; in practice there is a restriction to 40000 terms on the use of this routine.

Finally, one of a large number of compilers may be used to translate a given programme into basic machine instructions. All the programmes used to obtain the results in this thesis were written in a language whose compiler processes programmes written in both FORTRAN and ASP (the latter being almost basic machine code). Programmes written in ASP are apt to be more efficient than those written in FORTRAN, since FORTRAN may generate redundant instructions - from the programmer's point of view. Once compiled, both types of programme reside in binary form either on cards or on a subroutine library tape.

CHAPTER 2.

2.1.

This chapter describes computations concerning certain non-congruence subgroups of the classical modular group. A summary of the mathematics involved is given in section 2.2.; the complete mathematical background is to be found in Atkin and Swinnerton-Dyer (2). Some of the results arrived at would have required a prohibitive amount of time had they been obtained by hand calculation (i.e. pencil and paper). Indeed, it was only by the handling of these calculations by an electronic computer that they became at all feasible. Even so, as Atkin and Swinnerton-Dyer point out in their paper, the time involved is "not trivial in either human or machine terms". In terms of Atlas time (an average of 350,000 basic instructions per second), some of the computations required about an hour and it would have been quite possible to fabricate programmes needing several hours of computation, had one not had to consider the possibility of a machine failure from which recovery would have been difficult, if not impossible.

The last section of this chapter gives the (negative) results of statistical analysis on Ramanujan's τ-function and a few other related functions.

2.2. The classical modular group.

The classical, or full, modular group consists of all linear fractional transformations

$$V\tau = \frac{a\tau + b}{c\tau + d} \qquad (2.2.1)$$

where a,b,c and d are rational integers with ad-bc=1. This group, denoted

by Γ, is generated by the transformations S and T where

$$S\tau = \tau + 1 \quad , \quad T = -\frac{1}{\tau} \tag{2.2.2}$$

and is the free product of the cyclic groups $\{T\}$ and $\{P\}$ of order 2 and 3 respectively, where

$$P = TS \quad , \quad P\tau = -\frac{1}{\tau+1} \quad , \quad T^2 = P^3 = I \quad .$$

It is a discontinuous group whose fundamental domain F is shown as the shaded region in the figure in 3.6. The vertical sides of the boundary of F are mapped into each other by S and S^{-1} and the curved side into itself by T.

Klein's modular invariant, which is the Hauptmodul of Γ (see Klein - Fricke (11), p.591), is denoted by j where

$$j(\frac{a\tau + b}{c\tau + d}) = j(\tau) \quad , \quad ad - bc = 1 \quad .$$

j is zero at $\tau = \rho = e^{\frac{2\pi i}{3}}$, is equal to 1728 at $\tau = i$ and has a simple pole at $\tau = i\infty$.

If $x = e^{2\pi i\tau}$, then j may be written as a power-series in x

$$j = \sum_{n=-1}^{\infty} c(n)x^n = x^{-1} + 744 + 196884x + \ldots \tag{2.2.3}$$

where the coefficients c(n) are positive integers. These coefficients enjoy certain congruence properties. For example

if $$n \equiv 0 \pmod{2^a 3^b 5^c 7^d 11^e}$$

then $$c(n) \equiv 0 \pmod{2^{3a+8} 3^{2b+3} 5^{c+1} 7^d 11^e}.$$

From Atkin (1), this is probably the best possible congruence of this form.

We give below a table of c(n) for n=-1,...,40 with a corresponding approximation opposite each entry. For 'E' read 'times ten to the'.

c(n)	n	$\frac{e^{4\pi\sqrt{n}}}{\sqrt{2}\, n^{\frac{3}{4}}}$
1	-1	
744	0	
1 96884	1	2.0276E+05
214 93760	2	2.1968E+07
8642 99970	3	8.7967E+08
2 02458 56256	4	2.0557E+10
33 32026 40600	5	3.3776E+11
425 20233 00096	6	4.3050E+12
4465 69940 71935	7	4.5171E+13
40149 08866 56000	8	4.0581E+14
3 17644 02297 84420	9	3.2086E+15
22 56739 33095 93600	10	2.2784E+16
146 21191 14995 19294	11	1.4755E+17
874 31371 96857 75360	12	8.8197E+17
4872 01011 17981 42520	13	4.9130E+18
25497 82738 94105 25184	14	2.5704E+19
1 26142 91646 57818 43075	15	1.2713E+20
5 93121 77242 14450 58560	16	5.9761E+20
26 62842 41315 07752 45160	17	2.6824E+21
114 59912 78844 47865 13920	18	1.1542E+22
474 38786 80123 41688 13250	19	4.7768E+22
1894 49976 24889 33900 28800	20	1.9073E+23
7318 11377 31813 75192 45696	21	7.3663E+23
27406 30712 51362 46549 29920	22	2.7583E+24
99710 41659 93718 26935 33820	23	1.0034E+25
3 53074 53186 56142 70998 77376	24	3.5525E+25
12 18832 84330 42251 04333 51500	25	1.2262E+26
41 07899 60190 30790 91576 38144	26	4.1322E+26
135 35635 41518 64687 86750 77500	27	1.3614E+27
436 56892 24858 87663 46104 01280	28	4.3906E+27
1379 83758 34642 99992 55422 88376	29	1.3876E+28
4278 07822 44213 26256 70582 27200	30	4.3016E+28
13023 36938 25770 29512 80448 73221	31	1.3094E+29
38960 80061 70995 91189 43000 98560	32	3.9168E+29
1 14632 93989 00810 63777 96110 90240	33	1.1523E+30
3 31962 77091 39267 16726 36796 06784	34	3.3368E+30
9 46816 61357 02260 43164 62634 38600	35	9.5164E+30
26 61436 58257 53796 26887 21518 75584	36	2.6748E+31
73 77316 99697 25069 76080 17928 54360	37	7.4138E+31
201 76878 99472 28738 64858 00437 76000	38	2.0275E+32
544 76388 17516 16630 12316 54104 77688	39	5.4739E+32
1452 68925 44393 62169 79435 54293 76000	40	1.4596E+33

If $V'\tau = \frac{a'\tau + b'}{c'\tau + d'} \in \Gamma$, we write $V \equiv V'(\text{mod } a)$ if and only if $a \equiv a'$, $b \equiv b'$, $c \equiv c'$, $d \equiv d'(\text{mod } n)$. An important subgroup of Γ is the

set of all transformations

$$V \equiv \pm I \pmod{n}$$

where $I\tau = \tau$. It is called the principal congruence subgroup $\Gamma(n)$ of level n and is a normal subgroup of finite index in Γ. By taking n=1, Γ may be written as $\Gamma(1)$.

A subgroup G of finite index μ in Γ has a fundamental domain consisting of μ copies of F. If the elements of G conjugate in Γ to T and P form respectively e_2 and e_3 conjugacy classes in G, then the boundary of F will have e_2 and e_3 in equivalent fixed point vertices of orders 2 and 3 respectively. Let every element of G conjugate in Γ to a non-zero power of S be conjugate in G to some power of one of

$$S^{\mu_1}, \; g_2 S^{\mu_2} g_2^{-1}, \; \ldots, \; g_t S^{\mu_t} g_t^{-1}$$

where $g_1 = I, g_2, \ldots, g_t$ are in Γ and $g_i g_j^{-1} \notin G$. The boundary of F will then have t in-equivalent parabolic fixed point cusps. If t=1, we shall call G a cycloidal subgroup of Γ. We have

$$\mu = \mu_1 + \mu_2 + \cdots + \mu_t$$

and the genus g of G is given by

$$g = 1 + \frac{\mu}{12} - \frac{t}{2} - \frac{e_2}{4} - \frac{e_3}{3} \qquad (2.2.4)$$

Almost all subgroups of Γ are non-congruence subgroups in the sense that, if $C(\mu)$ and $N(\mu)$ denote respectively the numbers of congruence and non-congruence subgroups with index $\leq \mu$, then $C(\mu)/N(\mu) \to 0$ as $\mu \to \infty$.

If G is a subgroup of finite index in Γ then $f(\tau)$ is a modular form on G if

(1) $f(\tau)$ is single-valued and regular except for poles at the cusps of G

(2) $f(V\tau) = (c\tau + d)^{2\omega} f(\tau)$ for all V in G, where $V\tau = \frac{a\tau+b}{c\tau+d}$ and ω is a fixed integer.

When $\omega=0$, f is called a modular function on G; when $\omega = 1$, f is called a differential on G. In the former case, it can be shown that $f(\tau)$ is an algebraic function of $j(\tau)$ and that the only branch points of $f(\tau)$ are branch points of order 2 at which j=1728, branch points of order 3 at which j=0 and branch points at which j is infinite.

2.3. Γ_9 : a set of 9 non-congruence subgroups of level 9 and genus 1.

In Atkin and Swinnerton-Dyer (2), it is shown that a subgroup G of Γ of genus 0 can be completely specified by giving:-

(1) a specification for the group, i.e. a set of integers $\mu \geq 1$, $t \geq 1$, $e_2 \geq 0$, $e_3 \geq 0$, $\mu_1 \geq 1$ and, if $h > 1$, a set (a_i, v_i) with $a_i \geq 1$, $v_1 > v_2 > \ldots > v_s$, for i = 1 to s, satisfying

(2) relations between j and j-1728 and certain polynomials in ζ, the Hauptmodul of G - the "j - equations" of the specification;

(3) a relation between the coefficients of the polynomials in ζ derivable from the zero constant term in the expansion of ζ at ∞.

When $g > 0$, the situation is somewhat altered; from a consideration of branch points one still obtains "j - equations". In the case of the cycloidal subgroup Γ_9, where $t = 1$, $\mu = \mu_1 = 9$, $g = 1$, $e_2 = 1$ and $e_3 = 0$, the elimination of j from these equations produces $2\mu = 18$ simultaneous non-linear equations between as many unknowns whose solution effectively specifies Γ_9. If $x_1, x_2, \ldots, x_{18}$ denote the unknowns, the 18 equations are those obtained by equating the coefficients of x^i (i = 0 to 8) in both

the following identities :-

$$(x^3+x_1x^2+x_2x+x_3)(x^3+x_4x^2+x_5x+x_6)^2-(x_7x^4+x_8x^3+x_9x^2+x_{10}x+x_{11})^2$$
$$\equiv (x^3+x_{12}x^2+x_{13}x+x_{14})^3 \qquad (2.3.1)$$

$$(x^3+x_1x^2+x_2x+x_3)(x^3+x_4x^2+x_5x+x_6)^2-(x_7x^4+x_8x^3+x_9x^2+x_{10}x+x_{11}-1728)^2$$
$$\equiv x(x^4+x_{15}x^3+x_{16}x^2+x_{17}x+x_{18})^2 \qquad (2.3.2)$$

Thus the 18 equations are

$$\begin{aligned} x_3x_6^2-x_{11}^2 &= x_{14}^3 && \text{(coefficients of } x^0 \text{ in (2.3.1))} \\ &\vdots \\ x_2x_6^2+2x_3x_5x_6-2x_{10}(x_{11}-1728) &= x_{18}^2 && \text{(coefficients of } x^1 \text{ in (2.3.2))} \\ &\vdots \\ x_1+2x_4-x_7^2 &= 2x_{15} && \text{(coefficients of } x^8 \text{ in (2.3.2))} \end{aligned} \qquad (2.3.3)$$

Given the equations (2.3.3), the only information on the nature of the solutions that we were searching for was that the ninth powers of these solutions x_i ($i = 1$ to 18) were rational; more specifically, the x_i were of the form $\frac{m}{n}.k^r$, where k^9 was rational and m, n and r integral. By setting $x = \frac{y}{k^2}$ in (2.3.1) and (2.3.2) and multiplying throughout by k^{18}, the exponent r of k is uniquely determined for each solution. Thus we can determine integers s and t such that $\frac{x_i^s}{x_j^t}$ is rational.

The equations (2.3.3) were obtained by hand, though an algebraic manipulation package, had one been readily available, would have been used. This kind of algebra is best handled by a computer since by hand it is not only tedious but definitely unsafe. This does not imply that an electronic computer is 100% reliable; however, modern computers do have "hardware"

checks that trap almost all machine errors - see 1.3. Furthermore, an algebraic manipulation package could output directly to magnetic tape or disc and thus bypass the error-prone data preparation stage.

In the first instance, we attempted to solve the equations (2.3.3) directly, using a library programme designed to minimise a sum of squares of the form

$$F(x_1,x_2,\ldots,x_n) = \sum_{k=1}^{m}\{f_k(x_1,x_2,\ldots,x_n)\}^2 .$$

Here $m = n = 18$ and $f_1 \equiv x_3x_6^2 - x_{11}^2 - x_{14}^3, \ldots, \quad f_{18} \equiv x_1 + 2x_4 - x_7^2 - 2x_{15}$.

The programme demands that $x_1,\ldots,x_{18}$ must initially contain starting values for the search for solutions. Unfortunately, we were at a disadvantage in having no knowledge of the size of these solutions. There is, in fact, at least one infinite family of solutions and one isolated solution for the j - equations. We obtained from the computer such an isolated solution which was incorrect since the inequalities in the j - equations were not satisfied.

We give below the solutions x_i of (2.3.3), obtained by a completely different method. The solutions are of the form $\frac{m}{n}.k^r$, where

$$k = \left(\frac{3^3}{2^8}\right)^{1/9} = 0.7788578471 \ldots ;$$

the last column is an approximation to the solution.

i	m/n	r	x_i
1	225/4	2	34.12234
2	960	4	353.26778
3	4096	6	914.34300
4	54	2	32.75745
5	933	4	343.33212
6	5041	6	1125.29372
7	-9/2	1	-3.50486
8	-321	3	-151.66299
9	-16425/2	5	-2353.78278
10	-174303/2	7	-15152.42927
11	-306240	9	-32298.75000
12	48	2	29.11773
13	672	4	247.28744
14	2176	6	485.74472
15	72	2	43.67660
16	1872	4	688.87217
17	20544	6	4586.00164
18	81792	8	11075.83371

These solutions were obtained in the following way. On Γ_9 we have two functions

$$\left.\begin{aligned} x &= \xi^{-2} + \sum_{n=-1}^{\infty} a_1(n)\xi^n \\ \text{and} \quad y &= \xi^{-3} + \sum_{n=-1}^{\infty} a_2(n)\xi^n \end{aligned}\right\} \qquad (2.3.4)$$

where $\xi = e^{\frac{2\pi i \tau}{9}}$

Every function on Γ_9 is an algebraic function of x and y. Normally the coefficients $a_1(n)$ and $a_2(n)$ are complex; in the case of Γ_9, however, they can be taken to be real. By considering the pairing of the copies of the fundamental domain, it can be shown that if

$$\left.\begin{aligned} \tau_1 &= b_i + e^{i\theta} \\ \tau_2 &= c_i + e^{i(\pi-\theta)} \end{aligned}\right\} \qquad (2.3.5)$$

then $x(\tau_1)=x(\tau_2)$ and $y(\tau_1)=y(\tau_2)$ where $\frac{\pi}{3}<\theta<\frac{2\pi}{3}$ and the b_i, c_i are suitably chosen pairs of integers. On Γ_9 we have, for example, the pairings $b_1= 0, c_1= 0$; $b_2= 1, c_2= 4$; $b_3= 2, c_3= 6$ (amongst others). We now approximate

the theoretical situation by attempting to solve the simultaneous linear equations in the unknowns $a_1(n)$, $a_2(n)$ respectively, obtained by equating the coefficients in the expansions (2.3.4) at the points (2.3.5). Here θ assumes values in the closed interval $\left[\frac{\pi}{3}, \frac{2\pi}{3}\right]$ and the expansions (2.3.4) are now taken as finite expansions. The nett result of this device was the input to a linear programming package of two sets of 241 simultaneous linear equations in 40 unknowns. There was no guarantee of success since these equations were rather ill-conditioned. In the event, we were able to determine, from the solutions, the first 10 coefficients $a_1(n)$, $a_2(n)$ in the form $\frac{m}{m'}k^r$ (m and m' integral, $k^9 = \frac{27}{256}$), using a continued fraction expansion routine to identify m and m'.

By forming a new $y = y_{old} + \frac{3}{2}x$ and by fixing the constant terms in the expansions of x and y, we obtain the relations

$$\left.\begin{aligned} y^2 &= x^3 + 225/4k^2x^2 + 960k^4x + 4096k^6 \\ j &= y(x^3 + 54k^2x^2 + 933k^4x + 5041k^6) - \\ &\quad \tfrac{1}{2}(9kx^4 + 642k^3x^3 + 16425k^5x^2 + 174303k^7x + 612480k^9) \end{aligned}\right\} \qquad (2.3.6)$$

where $k^9 = \frac{27}{256}$. From these relations we can obtain the solutions of (2.3.3). Five minutes of computing time were required to set up and solve the simultaneous equations.

2.4. Congruence properties.

Having obtained the relations (2.3.6) for Γ_9, we may form the differential

$$\frac{dx}{y} = \frac{\xi dx}{2yd\xi} = \sum_{n=1}^{\infty} b(n)k^{n-1}\xi^n = \xi - k\xi^2 - 3k^2\xi^3 - k^3\xi^4 - 5k^4\xi^5 + \ldots$$

We give below a table of b(i), i=1(1)35,99(1)107. Each b(i) is of the form $\frac{m}{3^n}$, where $3 \nmid m$ (except for i=3,6 and multiples of 9). The right-hand entries against the primes p are the character sums $\sum_{x=0}^{p-1} \left(\frac{y^2}{p}\right)$.

r	n	i	character sum
1	0	1	
-1	0	2	1
-3	0	3	0
-1	0	4	
-5	0	5	0
3	0	6	
-5	0	7	-2
-1	0	8	
0	0	9	
-875	4	10	
274	4	11	3
539	2	12	
3932	4	13	-2
9365	4	14	
-910	2	15	
10159	4	16	
556	4	17	3
0	0	18	
-17 13787	8	19	1
37 94725	8	20	
75850	5	21	
-57 10514	8	22	
16 35149	8	23	6
2 50073	5	24	
-17680	8	25	
50 46244	8	26	
0	0	27	
-13695 44161	13	28	
-37955 93836	13	29	-6
-14 98630	5	30	
59133 93124	13	31	4
-1 98424 52435	13	32	
-202 63550	8	33	
-1 12065 50788	13	34	
-1 34310 87490	13	35	
⋮	⋮	⋮	
0	0	99	
-7755 40277 59242 84974 15239 83076 55920	48	100	
-62070 08185 13546 23220 61516 73050 03883	48	101	0
-2 46835 24201 03676 23262 10262	28	102	
-27813 30552 38924 86914 32633 14952 10611	48	103	-14
-85568 15354 28903 46806 04393 37025 91388	48	104	
-121 93450 57370 27048 51437 39040	32	105	
12140 91762 41078 27279 42019 39633 28412	48	106	
-1 94862 13591 18374 43392 49832 83215 38536	48	107	-3

To investigate the congruence properties of the b(i) and similar coefficients, it is desirable to have a subroutine which, given an elliptic curve $y^2 = x^3 + c(3)x^2 + c(2)x + c(1)$, a relation $j = y(d(N+1)x^N + \ldots + d(1)) + e(N+3)x^{N+2} + \ldots + e(1)$, and the value of k^μ, computes the coefficients in the expansions of $x = \xi^{-2} + \ldots$ and $y = \xi^{-3} + \ldots$ and the differential $-\frac{dx}{y} = \xi + \ldots$, all the computation being performed modulo a given integer. Here the c(i), d(i) and e(i) are rational and the index of the subgroup is μ where (2.4.1)

(1) if μ is even, $N = \frac{1}{2}(\mu-4)$ and $e(N+3) = 1$

(2) if μ is odd, $N = \frac{1}{2}(\mu-3)$, $d(N+1) = 1$, and $e(N+3) = 0$.

Thus, in the case of Γ_9, $\mu=9$ and N=3, with the relations (2.3.6) input to the subroutine. We may eliminate k from these relations using the transformations $x_{new} = x/k^2$, $y_{new} = y/k^3$ and the value $k^9 = \frac{27}{256}$.

Basically, the subroutine sets up arrays to contain the coefficients in the expansions of $x,\ldots,x^{N+2}$, $y,yx,\ldots,yx^N$ and j, the latter being calculated by KLEINJ. At any one time, two coefficients α in the expansion of x and β in the expansion of y are being determined as solutions of two simultaneous linear equations in the unknowns α and β, obtained by equating the relevant powers of ξ in the relations (2.4.1). All the arrays are then updated and the cycle continues until sufficiently many coefficients of x and y have been computed. The most expensive operation involved in this procedure is the use of a subroutine to multiply two power series, despite the fact that this latter subroutine employs a function that overwrites its "calling sequence" and plants the relevant code directly in the calling subroutine - see 4.3 .

Throughout the whole process, arithmetic is performed modulo a given integer which must be relatively prime to 2, μ and k^{μ}. It could be argued that a large number of coefficients of x and y and thus $\frac{dx}{y}$ should be calculated once and once only, using a multi-length version of the programme described above. This would avoid recomputing the coefficients each time a different modulus is required. Unfortunately, this is not feasible from the point of view of computing time. Thus, for Γ_9, the first 500 coefficients of $\frac{dx}{y}$ may be computed to a given modulus in approximately 30 seconds; to obtain these coefficients as rational numbers would, however, require several hours computing since a high precision would be necessary - see the previous table.

One of the basic congruence properties of the coefficients of $\frac{dx}{y}$ on Γ_9 is that, for p prime,

$$b(p) + \sum_{x=0}^{p-1} \left(\frac{y^2}{p}\right) \equiv 0 \pmod{p} .$$

In fact, a much more general result is known - see Atkin and Swinnerton-Dyer(2), 5.3.Theorem 4. Furthermore, from a mass of numerical evidence, we obtain the following properties. Given any "random" x and any x_2, x_1, x_0, we form y from

$$y^2 = x^3 + x_2x^2 + x_1x + x_0$$

and obtain $\frac{dx}{y}$ with coefficients a(n), say. Let $c(p) = \sum_{x=0}^{p-1} \left(\frac{y^2}{p}\right)$ and let Δ be the discriminant of the cubic $x^3+x_2x^2+x_1x+x_0$. Then, for integral k,

(1) if $p \nmid \Delta$ and $p \nmid c(p)$,

$$a(p^k n) + \lambda_k a(p^{k-1} n) \equiv 0 \pmod{p^k}$$

where $\lambda_k \equiv \lambda_{k-1} \pmod{p^{k-1}}$, $\lambda_1 \equiv -c(p) \pmod{p}$;

(2) if $p \nmid \Delta$ and $p \mid c(p)$,

$$a(p^k n) + \lambda_k a(p^{k-2} n) \equiv 0 \pmod{p^k}$$

where $\lambda_k \equiv p \pmod{p^k}$;

(3) if $p|\Delta$ and $p \nmid c(p)$,

$a(p^k n) + \lambda_k a(p^{k-1} n) \equiv 0 \pmod{p^k}$

where $\lambda_k \equiv -c(p) \pmod{p^k}$;

(4) if $p|\Delta$ and $p \mid c(p)$,

$a(p^k n) \equiv 0 \pmod{p^k}$.

It should be noted that a considerable amount of computation is required to check these four congruence properties, even for very small p, n and k.

2.5. Statistical analysis.

In this section we consider the elementary statistical analysis of certain functions involving Euler's series

$$f(x) = \prod_{m=1}^{\infty} (1 - x^m) = 1 - x - x^2 + x^5 + x^7 - \ldots$$
$$= 1 + \sum_{n=1}^{\infty} (-1)^n \left\{ x^{\frac{1}{2}n(3n-1)} + x^{\frac{1}{2}n(3n+1)} \right\} \qquad (2.5.1)$$

17,950 terms of the following four functions were computed (the restriction being due to the size of the three moduli involved):-

(1) $xf^{24}(x) = \sum_{n=1}^{\infty} F_1(n)x^n$, $F_1(n) = \tau(n)$, Ramanujan's τ-function

(2) $xf^{12}(x^2) = \sum_{n=1}^{\infty} F_2(n)x^n$, $F_2(2n) = 0$

(3) $xf^4(x)f^4(x^5) = \sum_{n=1}^{\infty} F_3(n)x^n$

(4) $xf^2(x)f^2(x^{11}) = \sum_{n=1}^{\infty} F_4(n)x^n$.

Each function was computed three times modulo three different moduli, namely 10^8-1, 10^8 and 10^8+1. $f^{12}(x)$ was obtained half-way through the computation of $f^{24}(x)$. These functions were obtained by successive

applications of the routine EULMUL - see 4.5. The last two functions required EULMUL and QEULMU. The running times for the four functions were approximately 21, 9, 8 and 5 minutes respectively. For each coefficient of x, p prime, the three integers were processed by a modified Chinese Remainder routine giving the value of that coefficient with "single-length" precision, i.e. approximately 11 significant decimals. Finally the values of $\theta_p^{(n)} = \cos^{-1}(F_n(p)/2p^{k_n/2})$, $0 \le \theta_p^{(n)} \le \pi$, were computed and stored on magnetic tape. Here $k_1=11$, $k_2=5$, $k_3=3$ and $k_4=1$. In all, there were $\pi(17950) = 2507$ values of $\theta^{(1)}$, $\theta^{(3)}$ and $\theta^{(4)}$ and $\pi(35900) = 3814$ values of $\theta^{(2)}$.

The object of these computations was to consider the distribution of the values of $\theta_p^{(n)}$, conjectured to be $\frac{2}{\pi}\sin^2\theta\, d\theta$ - see Cassels (5). Here we can only state a negative result. The numerical results of a χ^2-test with a 5% significance level and 199 degrees of freedom lead to the conclusion that one cannot reject the hypothesis that the $\theta_p^{(n)}$ do in fact satisfy the conjectured distribution.

Finally, we give below a table of the results of the analogous computation on Γ_9 with

$$-\frac{dx}{y} = \sum_{n=1}^{\infty} a(n)\xi^n$$

where, for example, $a(2) = -k$, $a(3) = -3k^2$, and $k^9 = 27/256$. The sample, however, was too small for a statistical analysis to be performed.

p	a(p)	$a(p)/2\sqrt{p}$
2	-0.77885785	-0.27536783
3	-1.81985864	-0.52534794
5	-1.83993637	-0.41142228
7	-1.11614136	-0.21093089
11	0.27787376	0.04189105
13	2.41894998	0.33544801
17	0.12586963	0.01526394
19	-2.90559048	-0.33329409
23	1.02015859	0.10635889
29	-2.17536941	-0.20197798
31	2.05592107	0.18462716
37	7.17638578	0.58989528
41	-1.28208539	-0.10011405
43	-5.12554484	-0.39081913
47	1.18831601	0.08666685
53	5.65045095	0.38807456
59	2.88925528	0.18807450
61	11.48031518	0.73495187
67	-5.97525614	-0.36499686
71	-2.17313009	-0.12895155
73	-2.60818903	-0.15263272
79	0.64310334	0.03617739
83	3.38551235	0.18580413
89	-4.15760741	-0.22035275
97	8.16229765	0.41437788
101	-10.88634463	-0.54161589
103	-2.95916217	-0.14576746
107	-7.62915465	-0.36876911
109	6.42390350	0.30764918
113	-0.98011192	-0.04610059
127	14.86671545	0.65960384
131	-2.78541510	-0.12168142
137	-7.79166490	-0.33284343
139	5.46060717	0.23158133
149	-13.36360276	-0.54739450
151	-3.71055400	-0.15098030
157	10.00373368	0.39919243
163	-2.89581236	-0.11340876
167	9.11865499	0.35281135
173	23.67556582	0.90000997
179	-1.16122622	-0.04339706
181	-7.72916906	-0.28725231
191	-6.72459301	-0.24328724
193	0.31685666	0.01140392
197	-8.14854663	-0.29027995
199	18.27192204	0.64763111

CHAPTER 3.

3.1. Accuracy of computation.

As stated in 1.5 , the Atlas computer stores numbers (and instructions) in lengths of 48 binary bits, called a word. All operations such as addition, subtraction, multiplication and division are performed on words although "small" integer arithmetic can be performed on half-words, or b-registers in Atlas terminology. All arithmetic is carried out in the accumulator, which is a double-length floating-point register with special circuitry. By "floating-point" we mean the representation of a number as a mantissa and an exponent, e.g. 3.716×10^{38}. By "fixed-point" we mean the representation of a number as a mantissa and a fixed exponent, e.g. 3299 (here the exponent of 10 is 0). In an Atlas word, 8 bits are reserved for the exponent and the remaining 40 for the mantissa. One bit in each section represents the sign, leaving 7 for the exponent and 39 for the mantissa. The nett result is that it is possible to represent positive or negative numbers whose magnitudes are approximately in the range 10^{-116} to 10^{113}, with a precision of 39 binary digits, or approximately 11 significant decimal digits. Taking into account errors arising from rounding-off after arithmetical operations, we can guarantee the accuracy of a computation to at most 10 significant decimal digits.

For most purposes, this accuracy is sufficient. There are, however, certain problems in pure mathematics which require very much greater accuracy and we shall see in section 2 how to achieve this using special routines designed for this purpose.

3.2. Variable-precision arithmetic.

Variable-precision, or multi-length, arithmetic is the name given to the process of linking together consecutive words in store in order to perform arithmetic to a precision greater than that available using only a single word. Routines for this purpose were written in the language Atlas Autocode by F.Lunnon of Manchester University - see under Compiler AB, section 2 of "Further Literature on Compilers AA and AB", University of Manchester Department of Computer Science. Most of the coding was in the format of basic machine instructions (Atlas Basic Language). These routines were adapted for use in the Hartran system on Atlas by M.Bird of the Atlas Computer Laboratory. They were written in ASP (Atlas Symbolic Programming Language) - see ICL(9). The routines are called from Fortran programmes (or, indeed, ASP programmes) by two subroutines MLO and MLT, whose arguments are functions which operate on multi-length variables. Thus, the Fortran statement

```
X = (Y + SQRTF(Z + A * B))/C                                    (3.2.1)
```

translates into the statement

```
CALL MLO (A,MULT(B),ADD(Z),SQRCOT,ADD(Y),DIV(C),TO(X))          (3.2.2)
```

where A,B,Z,Y,C and X are now multi-length variables and MULT, ADD, SQRCOT, DIV and TO are routines which respectively perform multiplication, addition, square-root taking, division and assignment to the specified precision. Any routine in Fortran can be so translated into a multi-length version of it. Variables are defined to be of precision P (in the sense that all operations performed on them have an accuracy of P significant decimal digits, not allowing for rounding-off errors) by calling the routine MLD. Thus, for

example, before statement (3.2.2) is used in a routine, the following statement must be executed:

CALL MLD (PREC(P),NAMES(X,Y,Z,A,B,C))

Only the mantissa section of each word is used in the space reserved for multi-length variables. Hence each word will contribute $\log_{10} 2^{39} \simeq 11.74$ decimal digits of accuracy. So a variable declared to a precision of 100,000 decimal digits will require approximately 100000/11.74 words of store, or about 17 blocks of store (1 block ≡ 512 words). Additional store is needed for the execution of various operations; the most expensive operation is SQROOT which requires 5 words of working store for each word of the multi-length variable in question. The multi-length routines thenselves occupy about 10 blocks of store, with the main "calculating" routine having over 2000 ASP instructions. For further details, see 4.5 .

The multi-length package is not so efficient that "organisational" overheads are negligible. For example, the Fortran statement Z = X + Y is executed in the Hartran system in three basic instructions, while the "equivalent" multi-length instruction CALL MLO (X,ADD(Y),TO(Z)) is executed in 544 instructions. This is an extreme case, however; as the precision increases, the ease with which multi-length instructions can be inserted into Fortran programmes to some extent outweighs the inefficiency of the computation. This latter observation is generally true of any high-level language, where macro-instructions replace a sequence of basic instructions.

Below is given a table indicating the performance of the original multi-length routines. The times taken by the Hartran multi-length package are in fairly good agreement with those obtained from this table. Each row

of the table contains the operation followed by a list of the number N of instructions obeyed for each "length" L in the top row. The precision P is given by $P = \log_{10} 2^{39} \times L \simeq 11.74L$ and the computing time T is given by $T \simeq N/327680$ seconds. Thus the time required to perform a division to a precision of 750 significant decimal digits is approximately $\frac{1}{5}$ second.

	1	2	4	8	16	32	64	128
ADD	320	350	400	490	710	1100	1800	3400
SUB	380	400	450	560	800	1300	2200	3900
MULT	320	400	650	1300	3600	12000	43000	160000
DIV	460	590	950	2100	5500	18000	62000	240000
POWER	3000	4400	7800	18000	49000	160000	570000	2100000
IPART	320	340	370	440	520	750	1100	2000
SQROOT	2300	3300	5000	8900	18000	44000	130000	430000
PRINT	6100	7200	10000	18000	47000	160000	570000	2200000
READ	8800	16000	32000	80000	240000	700000	3000000	11000000

Two further examples of computing times in the Hartran multi-length package are: (1) Square-root to a precision of 20,000 significant decimal digits in 2 minutes; (2) Addition to a precision of 200,000 significant decimal digits in 1.75 seconds.

The writer has used these routines to compute various exponentials and logarithms to a large number of significant digits. They were also used to calculate the partial quotients in the continued fraction expansion of certain numbers. An example is given in 3.4., which links the convergents of $\log \frac{(2+\sqrt{3})}{(3+\sqrt{8})}$ with the non-existence of solutions to a certain diophantine equation.

It is possible, and from a consideration of computing time, advisable, to avoid the wholesale use of multi-length arithmetic in certain problems dealing with rational numbers. This is discussed in the next section.

3.3. The Chinese Remainder Theorem and "modular" arithmetic.

It is often the case in number-theoretical calculations that the end result of a computation is a large integer or a sequence of these. Furthermore, the result is often known only up to equivalence modulo a particular small integer. It is intuitively evident that, if we have the answer in this form for sufficiently many moduli, then this information will be enough to determine the answer exactly or up to equivalence modulo a large integer. This is, in fact, the case. The Chinese Remainder Theorem states that every system of linear congruences in which the moduli are relatively prime in pairs is solvable, the solution being unique modulo the product of the moduli.

Let the system of congruences be

$$x \equiv a_i \pmod{m_i}, \quad i=1,2,\ldots,n \tag{3.3.1}$$

with $(m_i, m_j) = 1$ for $i \neq j$.

Put $M = \prod_{i=1}^{n} m_i$ and let $y = b_i$ be the solution of the congruence

$$\frac{M}{m_i} y \equiv 1 \pmod{m_i}.$$

Then the solution x of the system (3.3.1) is given by

$$x \equiv \sum_{i=1}^{n} a_i b_i \frac{M}{m_i} \pmod{M}.$$

The one disadvantage of this theorem is that one has to have some knowledge of the size of the numbers involved in a computation before that computation is carried out. When one does have such knowledge (and this is usually the case), then the combination of the Chinese Remainder Theorem and multi-length arithmetic is much more efficient than the use of multi-length arithmetic alone.

Applications of the Chinese Remainder Theorem will require a package

designed to carry out arithmetic modulo a given number. The routines written for this purpose are described in 4.5. These routines need not necessarily be used in conjunction with the Chinese Remainder Theorem.

As a simple example, a suitable number of executions of the routine EULDIV (see 4.5.) will set up in an array the coefficients c(k) of Klein's modular invariant $j = \sum_{k=-1}^{\infty} c(k)\, x^k$, computed modulo a given prime. From Lehmer (14), we know that $c(k) \sim \frac{e^{4\pi\sqrt{k}}}{\sqrt{2}\, k^{3/4}}$, which gives an approximate value for a particular c(k) - see 2.2. If j is now computed modulo sufficiently many distinct primes, the Chinese Remainder Theorem will give the exact value of c(k).

Several other applications of the Chinese Remainder Theorem are given in 3.5. and 3.6.

3.4. Continued fractions and the equations $3x^2-2 = y^2$ and $8x^2-7 = z^2$.

Let Θ be a positive real number. By its continued fraction expansion we shall mean the representation of Θ as

$$\Theta = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}} \qquad (3.4.1)$$

where the a_i are positive integers, called the partial quotients of Θ. The right-hand side of (3.4.1) will be written $[a_0; a_1, a_2, a_3, \ldots]$.

The a_i must be unique for they are obtained from the following algorithm. Let $r_0 = \Theta$, $a_0 = [r_0] = [\Theta]$, where $[x]$ is the largest integer $\leq x$, and let, for $n \geq 1$,

$$\left.\begin{aligned} r_n &= \frac{1}{r_{n-1} - a_{n-1}} \\ a_n &= [r_n] \end{aligned}\right\} \qquad (3.4.2)$$

The simplest kind of continued fraction expansion occurs when θ is either rational or a quadratic irrational. In the former case, the sequence $[a_0;a_1,a_2,\ldots]$ terminates; the denominator of r_n is zero for some n. In the latter case, the sequence of partial quotients is eventually periodic. When θ is neither rational nor a quadratic irrational, i.e. in the case when θ is algebraic of degree ≥ 3, or transcendental, very little is known about the sequence of partial quotients. A notable exception is e, whose partial quotients form the sequence $[2;1,2,1,1,4,1,\ldots,1,2n,1,\ldots]$.

If in (3.4.1), the sequence is terminated at a particular a_n, the resulting

$$\theta' = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 \cdots + \cfrac{1}{a_n}}}} = \frac{p_n}{q_n} \tag{3.4.3}$$

will be an approximation to θ; the larger n is, the more accurate is the approximation θ' to θ. These rational approximations are called the convergents to θ and can be obtained from the partial quotients $a_0,a_1,a_2,\ldots$ by the following algorithm.

Let $p_{-1} = q_{-2} = 1$ and $p_{-2} = q_{-1} = 0$.

Then for $n > 0$

$$p_n = a_n p_{n-1} + p_{n-2} \quad \text{and} \quad q_n = a_n q_{n-1} + q_{n-2} \tag{3.4.4}$$

The elementary properties of convergents are:

(1) For $n \geq 0$, $q_n p_{n-1} - p_n q_{n-1} = (-1)^n$ (3.4.5)

(2) For $n \geq 0$, $\left|\theta - \frac{p_n}{q_n}\right| < \frac{1}{q_n q_n + 1}$

(3) For $n \geq 0$, $(p_n,q_n) = 1$.

Gauss first posed, in 1812, the problem that led eventually to the "metric" theory of continued fractions; see Khinchin (10). This theory is concerned with those properties of the continued fraction expansion which are true for almost all real numbers, i.e. for all numbers with the exception of a set of measure zero. We consider Θ to satisfy $0 \leq \Theta < 1$; it is clear from the definition of the a_i that the continued fraction expansion of Θ is essentially the same as that of $\Theta+m$, where m is an arbitrary integer.

Hence we set

$$\Theta = [0; a_1, a_2, a_3, \ldots]$$

and have

$$r_n = [a_n; a_{n+1}, a_{n+2}, \ldots].$$

Let $z_n = r_n - a_n$. We have $0 \leq z_n < 1$. Gauss stated that he had proved that

$$\text{Prob}\,(z_n < x) = \log_2(1+x).$$

His proof was never published; it was, in fact, first proved by Kusmin in 1928 - see Kusmin (12).

Some properties obtained by the "metric" theory are:

(1) For almost all Θ , $\text{Prob}(a_i = k) = \log_2 \frac{(k+1)^2}{k(k+2)}$ (3.4.6)

(2) For almost all Θ , $\lim_{n\to\infty} \sqrt[n]{q_n} = e^{\frac{\pi^2}{12\log 2}} = 3.27582\ 29187\ 21811\ 15978\ldots$ (3.4.7)

(3) For almost all Θ , $\lim_{n\to\infty} \sqrt[n]{a_1 a_2 \ldots a_n} = K$, where K is Khinchin's constant

$$\prod_{r=1}^{\infty} \left\{1 + \frac{1}{r(r+2)}\right\}^{\log_2 r} = 2.68545\ 20010\ 65306\ 44530\ \ldots.$$

(1) and (3) are due to Khinchin (10) ; (2) is due to Levy (15).

We now consider computations in connection with a result of Baker and Davenport (4). In that paper, Baker and Davenport discuss the following problem. The four numbers 1,3,8 and 120 have the property that the product of any two, increased by 1, is a perfect square. They prove that

the number 120 cannot be replaced by any other integer $N>0$, if the same property is to hold. The best result that had been obtained previously was that of Professor J.H.van Lint (23), who showed that if N existed,

$$N > 10^{1700000} \tag{3.4.8}$$

The problem reduces to showing that the simultaneous equations

$$\left.\begin{aligned} 3x^2 - 2 &= y^2 \\ 8x^2 - 7 &= z^2 \end{aligned}\right\} \tag{3.4.9}$$

have no solutions in positive integers, other than x=1 and x=11. As Baker and Davenport point out in their article, Siegel's theorem (see Siegel (17), also indexed under X (24)) may be applied to the equation

$$(3x^2 - 2)(8x^2 - 7) = y^2z^2 = t^2 \tag{3.4.10}$$

This theorem shows that (3.4.10) can have only finitely many integer solutions, but, as it depends ultimately on Thue's theorem on diophantine equations, it offers no possibility of determining an explicit upper bound for x satisfying (3.4.10).

However, Baker (3) has proved a theorem, whose application to the equations (3.4.9) yields the following result. If

$$(2\sqrt{3})x = (1 + \sqrt{3})(2 + \sqrt{3})^m - (1 - \sqrt{3})(2 - \sqrt{3})^m \tag{3.4.11}$$

then any solution x of (3.3.6) must satisfy

$$m < (4^{15}\log 2880)^{49} < 10^{487}.$$

Since the solution x=11 of (3.4.9) corresponds to m=2 in (3.4.11), the range in which further solutions may be found is

$$2 < m < 10^{487} \tag{3.4.12}$$

A direct search for solutions of (3.4.9) in this range would have been quite impossible in the light of present-day computing power. This

range can, however, be almost completely eliminated by the application of a lemma given by Baker and Davenport (4). The application shows that if

(1) $\theta = \dfrac{\log(2+\sqrt{3})}{\log(3+\sqrt{8})} = 0.74710\ 53797\ 84665\ 20012\ \ldots$

and $C = (2+\sqrt{3})^2 = 13.92820\ 32302\ 75509\ 17410\ \ldots$

(2) $\beta = \dfrac{\log \frac{(1+\sqrt{3})\sqrt{8}}{(1+\sqrt{8})\sqrt{3}}}{\log(3+\sqrt{8})} = 0.08680\ 37805\ 12726\ 74666\ \ldots$

and $\beta' = \dfrac{\log \frac{(1+\sqrt{3})\sqrt{8}}{(\sqrt{8}-1)\sqrt{3}}}{\log(3+\sqrt{8})} = 0.50603\ 46008\ 68222\ 91804\ \ldots$

(3) $M = 10^{487}$ and $K = 10^{33}$

(4) θ_0 = value of θ correct to 1040 decimal places,
i.e. $|\theta - \theta_0| < 10^{-1040}$

(5) q is the denominator of the last convergent, in the continued fraction expansion of θ_0, satisfying $q < 10^{520}$

(6) $\|q\beta\| \geqslant 3 \times 10^{-33}$ and $\|q\beta'\| \geqslant 3 \times 10^{-33}$, where, for real x, $\|x\| = |x - [x + \frac{1}{2}]|$, i.e. $\|x\|$ is the absolute value of the difference of x and the nearest integer to x;

then there is no solution of (3.4.9) with m in the range

$$\frac{\log K^2 M}{\log C} < m < M$$

i.e. $$\frac{\log 10^{553}}{2\log(2+\sqrt{3})} < m < 10^{487} \qquad (3.4.13)$$

The number on the left-hand side is approximately 483.5.

The main computation is that involved in finding q in (5). To find θ, β, β' the author used the square-root facility already in the Hartran multi-length package and his own logarithm routine. This latter routine simply uses the standard series for log(1 + x) namely

$$\log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots + (-1)^{n+1}\frac{x^n}{n} + \ldots \qquad (-1 < x \leqslant 1) \qquad (3.4.14)$$

If $R_N(x) = \left| \log(1+x) - \sum_{n=1}^{N} \frac{(-1)^{n+1}x^n}{n} \right|$, then

$$R_N(x) < \frac{x^{N+1}}{N+1} \quad \text{if} \quad x > 0$$

$$\text{and } R_N(x) < \frac{(-x)^{N+1}}{1+x} \quad \text{if} \quad x < 0.$$

An approximation y is taken to the number Y, whose logarithm is required. Let l be the single-length logarithm of y, i.e. log y, evaluated to a precision of approximately 11 significant decimal digits. Let E = multi-length exponential of l. The series (3.4.14) is then used with x = Y/E.

Wrench and Shanks (22) have written on the effective computation of the continued fraction expansion of a real number. In most cases, their method will produce the desired results, although provision does have to be made for the possibility that very large partial quotients may occur in the expansion. This procedure was initially adopted but the computing overheads in Fortran multi-length computation (see 3.2.) made it only marginally more efficient than the use of the algorithm (3.4.2) ; moreover, with this algorithm no special provision need be made for very large partial quotients.

The denominators q_n of the convergents were computed by(3.4.4) and the denominator q of the last convergent satisfying $q \leqslant 10^{520}$, namely q_{1002}, found to be

74766 56458 85928 21002 92900 19462 74193 99932
88435 51834 20544 67033 92527 99010 36030 14382 83128 15409 94079 49641
75823 72448 20294 43561 15091 97552 65496 09837 65725 70805 71781 03765
90201 82968 04828 89690 91216 09036 42656 74598 43126 05161 50601 13889
48311 34448 43630 77762 01995 69513 73885 70540 20065 08420 17453 43949
32542 08937 08733 92823 67336 28270 20008 54767 81468 64873 46464 28193
39455 78382 27505 86507 22688 57730 19978 42556 32569 44952 91835 82629
52538 66886 97685 22768 40839 96403 83429 92464 53306 46774 48258 60409
41197 29139 39485 18564 04207 26381 80339 63053 74225 67257 33135 04814

We remark that, referring to the asymptotic relation (3.4.7), the solution for x of the equation

$$e^{\frac{\pi^2 x}{12 \log 2}} = 10^{520}$$

is found to be $x \simeq 1009.08$, in surprisingly good agreement with the result

$$q_{1002} \simeq 7.477 \times 10^{519}.$$

Multiplying q by β and β' respectively, and taking the fractional part, we obtain

$$\| q\beta \| = 0.42279\ 62795\ 75983\ 04235\ \ldots$$
$$\text{and } \| q\beta' \| = 0.47422\ 86563\ 98614\ 56344\ \ldots$$

Thus (6) holds with a comfortable margin and this disposes of the range (3.4.13). The possibility of solutions in the range

$$2 < m < 483.5$$

is ruled out by the result (3.4.8).

3.5. The equation $x^3 - 6x^2 + 4x - 2 = 0$.

This section contains the results of computer-aided investigations of the real root of the cubic

$$x^3 - 6x^2 + 4x - 2 = 0 \qquad (3.5.1)$$

These investigations were carried out to explain the occurrence of very large partial quotients in the continued fraction expansion of this root. Part of this section appears in Churchhouse and Muir (6). All the variable-precision work was performed using the Hartran multi-length arithmetic package described in 3.2 .

The equation (3.5.1) was originally studied by D.H.Lehmer as the equation

$$y^3 - 8y - 10 = 0$$

obtained by setting $x = y + 2$ in (3.5.1). Before this, Delone and Faddeev (7) had made a study of the cubics

$$x^3 - ax - b = 0 \qquad a,b \text{ integral and } |a|, |b| \leqslant 9.$$

Unlike rational numbers and quadratic irrationals, very little is known about the structure of the continued fraction expansion of the roots of cubic or higher order algebraic equations. The continued fraction expansion of a rational number must eventually terminate and the continued fraction expansion of a quadratic irrational $\frac{a + b\sqrt{d}}{c}$ (a,b,c,d integral, d not a perfect square) has a periodic structure, the length of the period depending on d. Lehmer observed, however, that eight large partial quotients occurred within the first 200 partial quotients in the continued fraction expansion of the real root 3.31862 82177 ... of the cubic $x^3 - 8x - 10 = 0$. The smallest of these was 22986 and the largest 16467250. This is unusual for the following reason. From (3.4.6), for almost all real θ, the probability that in the continued fraction expansion of θ a particular partial quotient a_i equals k is $\log_2 \frac{(k+1)^2}{k(k+2)}$.

Hence, for almost all θ,

$$\text{Prob}(a_i \geqslant k) = \sum_{m=k}^{\infty} \log_2 \frac{(m+1)^2}{m(m+2)} = \log_2\left(1+\frac{1}{k}\right) \sim \frac{1.44}{k}.$$

Thus the probability that any particular partial quotient has a value greater than 20,000 is approximately $\frac{1}{13890}$. So the occurrence of eight such partial quotients amongst the first 200 partial quotients in the continued fraction expansion of the same algebraic number must be regarded as unusual.

The crucial step in the explanation of this phenomenon is the recognition that the equation $x^3 - 6x^2 + 4x - 2 = 0$ is precisely the equation satisfied by a class-invariant of Weber's, namely $f(\sqrt{-163})$.

Let j denote Klein's modular invariant (see 2.2.) and let

$$f(\tau) = q^{-\frac{1}{24}} \prod_{n=1}^{\infty} (1 + q^{2n-1})$$

where $q = e^{\pi i \tau}$.

To illustrate the theory of class-invariants constructed by Weber (see Weber(20)), let -n be a negative integer with $n \equiv 3 \pmod 8$. Let h be the class-number of the determinant -n and let

$$a_k x^2 + b_k xy + c_k y^2 \qquad (k = 1,2,\ldots,h)$$

be a complete set of reduced imprimitive binary quadratic forms with determinant $b_k^2 - 4a_k c_k = -n$. To construct such a set, one takes $b = \pm 1, \pm 3, \pm 5, \ldots$ with $|b| < \sqrt{\frac{1}{3}n}$ and then expresses $\frac{1}{4}(b^2+n)$ as a product of positive factors a and c in all possible ways with either $c > a$ and $-a < b \leq a$ or $c = a$ and $0 \leq b \leq a$. Clearly the form

$$x^2 + xy + \tfrac{1}{4}(n+1) \tag{3.5.2}$$

will belong to this set.

Let now τ_k be that root, with positive imaginary part, that satisfies the equation

$$a_k x^2 + b_k x + c_k = 0.$$

Then, by Weber (20), 418-423, the equation

$$\prod_{k=1}^{h} (x - j(\tau_k)) = 0 \tag{3.5.3}$$

has integral coefficients.

Furthermore, we have

$$j\left(\frac{-1+\tau}{2}\right) = -\frac{(f^{24}(\tau) - 256)^3}{f^{48}(\tau)}.$$

From (3.5.2), one of the τ_k must be $\frac{-1+\sqrt{-n}}{2}$. Hence $f^{24}(\sqrt{-n})$ must satisfy

an algebraic equation of degree 3h since, from (3.5.3), $j(\frac{-1+\sqrt{-n}}{2})$ satisfies an algebraic equation of degree h. It can be shown that this equation of degree 3h reduces to an equation

$$\prod_{k=1}^{3h}(x-\alpha_k)=0 \qquad (3.5.4)$$

with integral coefficients where the 24th powers of the roots α_k are the roots of the equation satisfied by $f^{24}(\sqrt{-n})$. Hence $f(\sqrt{-n})$ satisfies an algebraic equation of degree 3h, namely (3.5.4).

To illustrate the above theory, we take the discriminant $D = -2347$. The class-number $h(-2347) = 5$. By (3.5.3), $y = j(\frac{-1+\sqrt{-2347}}{2})$ will satisfy a quintic equation, namely

$$y^5 + 12\ 54200\ 00889\ 28251\ 61202\ 59144\ 98103\ 48856\ 55669\ 55018\ 77957\ 72767\ 51093\ 76000y^4 + 38503\ 92729\ 35672\ 30897\ 76031\ 37789\ 72411\ 73629\ 20974\ 45730\ 99844\ 79353\ 24160\ 00000y^3 - 397\ 49801\ 64212\ 09870\ 83534\ 73429\ 08155\ 23209\ 04452\ 72713\ 27833\ 29325\ 08876\ 80000\ 00000y^2 - 275\ 00560\ 58932\ 25872\ 28352\ 89220\ 63608\ 43069\ 45561\ 56276\ 88517\ 21342\ 43991\ 55200\ 00000\ 00000y + 9\ 01644\ 56992\ 25621\ 59908\ 07505\ 26731\ 94391\ 59181\ 35938\ 85976\ 89068\ 08033\ 28000\ 00000\ 00000\ 00000 = 0,$$

with discriminant

$$2^{324}.3^{132}.5^{62}.7^{40}.11^{22}.13^{20}.23^{8}.29^{12}.43^{12}.59^{8}.61^{2}.71^{4}.73^{2}.83^{4}.89^{4}.103^{4}:$$
$$107^{2}.109^{2}.127^{4}.157^{2}.167^{2}.197^{2}.199^{2}.227^{4}.239^{4}.251^{4}.263^{2}.269^{4}.281^{2}.293^{2}.359^{2}.$$
$$383^{2}.409^{2}.431^{2}.439^{2}.461^{2}.467^{2}.509^{2}.647^{2}.1103^{2}.1231^{2}.1259^{2}.1319^{2}.1499^{2}.$$
$$1583^{2}.1663^{2}.1823^{2}.2347^{2}\ .$$

The factorisation of the discriminant, which is approximately 4.98×10^{575}, required 15 seconds of computing; for a "random" number of this magnitude, the problem of its factorisation remains intractable for the forseeable future.

Similarly by (3.5.4), $x = f(\sqrt{-2347})$ will satisfy an equation of degree 15, namely

$$x^{15} - 566x^{14} - 950x^{13} - 2x^{12} + 1676x^{11} + 1688x^{10} - 1216x^9$$
$$- 5080x^8 - 3520x^7 + 1136x^6 + 3056x^5 + 5888x^4 - 3728x^3$$
$$- 9152x^2 - 224x - 32 = 0 \quad .$$

The discriminant of this equation is

$$-2^{102}.3^{28}.5^6.7^4.13^{10}.43^2.73^2.109^2.409^2.2347^7$$

and the real root is

$$567.67349\ 42228\ 67666\ \ldots = e^{\frac{1}{24}\pi\sqrt{2347}} \prod_{n=1}^{\infty} (1 + e^{-(2n-1)\pi\sqrt{2347}}) \quad .$$

We now fix $D = -d = -163$. Here the class-number $h(-163) = 1$, with $y = j(\frac{-1 + \sqrt{-163}}{2})$ satisfying the linear equation

$$y + (2^6.3.5.23.29)^3 = 0 \qquad (3.5.5)$$

The constant term in this equation is a perfect cube; indeed if $p > 0$ is a prime and $h(-p) = 1$, both $j(\tau)$ and $\sqrt[3]{j(\tau)}$ are integers when $\tau = \frac{-1 + \sqrt{-p}}{2}$. The equation satisfied by $x = f(\sqrt{-163})$ is a cubic, namely

$$x^3 - 6x^2 + 4x - 2 = 0$$

with discriminant $-2^2.163$.

This is precisely the equation Lehmer had investigated. From the above theory, we know that its real root, which we shall denote by Θ, is

$$e^{\frac{1}{24}\pi\sqrt{163}} (1 + e^{-\pi\sqrt{163}})(1 + e^{-3\pi\sqrt{163}})\ldots \qquad (3.5.6)$$

The main factor of this expression is $e^{\frac{1}{24}\pi\sqrt{163}}$, which approximates Θ with an error of less than 10^{-17}. The continued fraction expansion of $e^{\frac{1}{24}\pi\sqrt{163}}$ is unremarkable in the sense that no "large" partial quotients appear early on in the expansion. The remainder of the expression (3.5.6) involves the odd powers of $e^{\pi\sqrt{163}}$, which we shall denote by X. By (3.5.5)

$$j\left(\frac{-1+\sqrt{-163}}{2}\right) = -(2^6.3.5.23.29)^3 = N \quad , \quad \text{say.}$$

From the definition of j,

$$j(\tau) = \sum_{n=-1}^{\infty} c(n)x^n = x^{-1} + 744 + 196884x + 21493760x^2 + \dots ,$$

where

$$x = e^{2\pi i\tau} .$$

Setting $\tau = \frac{-1+i\sqrt{163}}{2}$, we obtain

$$x = e^{\pi i(-1+i\sqrt{163})} = -e^{-\pi\sqrt{163}} = -X^{-1}$$

and

$$N = -e^{\pi\sqrt{163}} + 744 - 196884e^{-\pi\sqrt{163}} + 21493760e^{-2\pi\sqrt{163}} - \dots$$

Thus

$$e^{\pi\sqrt{163}} = -N + 744 - 196884e^{-\pi\sqrt{163}} + 21493760e^{-2\pi\sqrt{163}} - \dots \qquad (3.5.7)$$

Since, by Lehmer (2),

$$c(k) \sim \frac{e^{4\pi\sqrt{k}}}{2^{\frac{1}{2}}.k^{\frac{3}{4}}} \qquad \text{as} \quad k \to \infty$$

it follows that the coefficients of the powers of $e^{-\pi\sqrt{163}}$ are heavily outweighed by these negative exponentials, so that, from (3.5.7), $e^{\pi\sqrt{163}}$ must be nearly an integer, the error being approximately $-196884e^{-\pi\sqrt{163}}$. In fact, computation shows that

$$X = e^{\pi\sqrt{163}} = 262\ 53741\ 26407\ 68743.99999\ 99999\ 99250 \dots$$

We now show that X^2 is nearly an integer; which is not trivially obvious. Let $M = -N + 744$. From (3.5.7)

$$X = M - 196884X^{-1} + 21493760X^{-2} - \dots$$

Hence, multiplying both sides by X,

$$X^2 = MX - 196884 + 21493760X^{-1} - \dots$$

$$= M(M-196884X^{-1}+21493760X^{-2}-\dots) - 196884 + 21493760X^{-1} - \dots$$

$$= M^2 + M(-196884X^{-1}+21493760X^{-2}-\dots) - 196884 + 21493760X^{-1} - \dots$$

$$= M^2 + (X+196884X^{-1}-21493760X^{-2}+\ldots)(-196884X^{-1}+21493760X^{-2}-\ldots)$$
$$- 196884 + 21493760X^{-1} - \ldots$$
$$= M^2 - 393768 + 42987520X^{-1} + O(X^{-2})$$

We therefore deduce that $e^{2\pi\sqrt{163}}$ should differ from an integer by approximately $42987520/262537412640768744 \simeq 1.63739 \times 10^{-10}$, the error being one of excess.

In a similar manner one could deal with $e^{3\pi\sqrt{163}}$, $e^{4\pi\sqrt{163}}$, ... It is found that the first 8 powers of $e^{\pi\sqrt{163}}$ are nearly integers, the error increasing fairly rapidly. The fractional parts of the first 9 powers of X are shown below.

Power of X	Fractional part
X	.99999 99999 99250 ...
X^2	.00000 00001 63738 ...
X^3	.99999 99901 23693 ...
X^4	.00000 03084 64322 ...
X^5	.99999 36541 87468 ...
X^6	.00009 71752 54162 ...
X^7	.99880 93165 26134 ...
X^8	.01223 41690 69154 ...
X^9	.29886 26339 54035 ...

The above analysis shows that, in particular, the first few odd powers of $X = e^{\pi\sqrt{163}}$ are nearly integers. Now, as observed previously, the continued fraction expansion of $e^{\frac{1}{2}\pi\sqrt{163}}$ is unremarkable and indeed it is found that the values of the partial quotients first differ from the partial quotients of Θ at the 16th. term, immediately before the first large term occurs. This implies that the first factor ignored, namely

$$1 + e^{-\pi\sqrt{163}} \qquad (3.5.8)$$

is, in some sense, responsible for the first large term in the continued

fraction expansion of Θ. Now (3.5.8) may be written as

$$1 + \frac{1}{M_1+}\ \frac{1}{1+}\ \frac{1}{N_1+} \cdots$$

where $M_1 = 262\ 53741\ 26407\ 68743$

and $N_1 = 133\ 34624\ 07511$.

The second factor ignored is $1 + e^{-3\pi\sqrt{163}}$ and this is nearly an integer, so that it may be written as

$$1 + \frac{1}{M_2+}\ \frac{1}{1+}\ \frac{1}{N_2+} \cdots$$

where $M_2 \simeq 1.8 \times 10^{52}$ and $N_2 \simeq 10^8$.

Similar remarks apply to the third and fourth factors, the corresponding values of M_i and N_i being

$$M_3 \simeq 1.24 \times 10^{87}\ , \quad N_3 \simeq 1.56 \times 10^5$$
$$M_4 \simeq 8.6 \times 10^{121}\ , \quad N_4 \simeq 800\ .$$

To test the hypothesis that the very large partial quotients in the continued fraction expansion of Θ are caused by the presence of these unusual factors which contain two large integer terms separated by a single 1, we compute the continued fraction obtained by taking N_1, N_2, N_3 and N_4 to be infinite. This is achieved in practice by replacing (3.5.6) by

$$\Theta_1 = X^{\frac{1}{24}}(1 + z_1^{-1})(1 + z_3^{-1})(1 + z_5^{-1}) \cdots \qquad (3.5.9)$$

where $z_n = \left[X^n + \frac{1}{2}\right]$, so that z_n is the nearest integer to X^n. The value of Θ_1 so obtained is extremely close to Θ, the error being approximately 5.8×10^{-47}. Despite this minute numerical change, the effect on the continued fraction expansion of Θ is so drastic that only the first large term remains and the remaining large partial quotients disappear. When the

factors in (3.5.9) are replaced, one by one, by their correct values in (3.5.6), the large partial quotients reappear one or two at a time.

We may thus summarise the reasons why the root of the cubic $x^3 - 6x^2 + 4x - 2 = 0$ has a remarkable continued fraction:

(1) the equation is that equation satisfied by $f(\sqrt{-163})$;

(2) the field $R(\sqrt{-163})$ obtained by adjoining $\sqrt{-163}$ to the rationals has class-number 1;

(3) the root Θ of the equation is approximated to seventeen places of decimals by $\Theta' = e^{\frac{1}{2}\pi\sqrt{163}}$;

(4) the ratio Θ'/Θ is given by the infinite product $\prod_{n=1}^{\infty}(1 + e^{-(2n-1)\pi\sqrt{163}})$;

(5) the first few terms of this product may all be written in the form
$$1 + \frac{1}{M_i +} \; \frac{1}{1 +} \; \frac{1}{N_i +} \cdots$$
where M_i is "very large" and N_i is "large";

(6) the presence of such factors in (5) produces large partial quotients in the continued fraction expansion of Θ. When these factors are replaced by factors of the type given in (3.5.9), the large terms disappear although the resulting change in the value of Θ is extremely small.

On the basis of these observations we can make a prediction. We have seen that $e^{n\pi\sqrt{163}}$ is very nearly an integer for $n = 1,2,\ldots,8$, the closeness of the approximation decreasing from about 10^{-12} when n=1 to about 10^{-2} when n=8. Hence the remark made at (5) above should not apply from n=9 onwards. Thus there is no reason to expect the large partial quotients

in the continued fraction expansion of Θ to persist beyond about the 170th. term, the point where the factor $1 + e^{-9\pi\sqrt{163}}$ could be expected to have an effect. This prediction is borne out. A computation of the first 875 partial quotients of Θ reveals no "large" terms after the one shown at position 161 in the table at the end of this section.

The large partial quotients in the continued fraction expansion of Θ do not themselves exhibit any remarkable arithmetical properties. An arithmetical property can, however, be obtained in the following manner. The convergents of Θ form an infinite sequence

$$\frac{5}{1}, \frac{16}{3}, \frac{117}{22}, \frac{484}{91}, \frac{1085}{204}, \ldots \qquad (3.5.10)$$

Evaluating the expression $x^3 - 6x^2 + 4x - 2$ where x is a convergent in (3.5.10), we obtain the following sequence of rational numbers

$$-\frac{7}{1}, \frac{10}{3^3}, -\frac{119}{22^3}, \frac{1002}{91^3}, -\frac{163}{204^3}, \ldots \qquad (3.5.11)$$

in which every other member has the same sign (which follows from the theory of convergents) and each member is of the form $\frac{m}{\pm n^3}$, m and n integral and $m > 0$, since Θ is irrational. Let m_{i-1} be the numerator of the i-th. member of the sequence (3.5.11). We give below a table of m_i, $i=0,1,\ldots,37$, together with the corresponding partial quotient a_i of Θ.

m_i	a_i	i
7	5	0
10	3	1
119	7	2
1002	4	3
163	2	4
1 34802	30	5
17381	1	6
3 82402	8	7
24 27371	3	8
22 25955	1	9
62 10806	1	10
16 23023	1	11
637 75085	9	12
1026 38689	2	13
4716 76211	2	14
3099 14182	1	15
1 85801	3	16
3305 64995 92126	22986	17
12823 92240 92853	2	18
898 30548 74131	1	19
1 14854 71083 26485	32	20
25 63788 06077 75803	8	21
102 88698 97801 28499	2	22
27 75517 62931 32362	1	23
37 44439 84778 89699	8	24
96222 24689 62929 21027	55	25
18163 68699 49053 77558	1	26
3 19471 73483 92623 08047	5	27
51995 08920 34725 81190	2	28
361 01815 94536 22961 64927	28	29
64 47213 54090 76471 05591	1	30
2294 80934 02128 87382 74802	5	31
208 32262 06868 99807	1	32
34866 85461 91617 91885 89642 67242	1501790	33
12117 74075 81146 78024 58582 07401	1	34
95682 73335 49158 14828 46042 02582	2	35
23781 13162 58472 58147 83317 18325	1	36
2 18311 24289 94519 03771 93404 67313	7	37

The m_i are certainly not monotonically increasing, but one observes a distinct fall in the value just before the position corresponding to one of the large partial quotients in the continued fraction expansion of Θ. One can expect this, since, at this position, we have a convergent

which, in relation to the size of its denominator, is a good approximation to Θ.

We observe two properties of the sequence $m_0, m_1, m_2, \ldots$. Firstly, the number 163 appears amongst it. Secondly, the eight "large" partial quotients a_i are

$$\begin{aligned} a_{17} &= 22986 \\ a_{33} &= 1501790 \\ a_{59} &= 35657 \\ a_{81} &= 49405 \\ a_{103} &= 53460 \\ a_{121} &= 16467250 \\ a_{139} &= 48120 \\ a_{161} &= 325927 \end{aligned}$$

The corresponding values of m_i begin

$$\begin{aligned} m_{16} &= 1\ 85801 \\ m_{32} &= 208\ 32262\ 06868\ 99807 \\ m_{58} &= 34354\ 87084\ 12692\ 17891\ 33839\ 29164\ 09801 \end{aligned}$$

For brevity, the approximate values of the remaining five m_i are

$$m_{80} \simeq 1.28 \times 10^{49}, \quad m_{102} \simeq 6.88 \times 10^{63}, \quad m_{120} \simeq 7.72 \times 10^{74}, \quad m_{138} \simeq 2.97 \times 10^{93}, \quad m_{160} \simeq 1.02 \times 10^{107}.$$

The attempted factorisation of these numbers reveals that

$$\begin{aligned} m_{16} &= 7 \cdot 11 \cdot 19 \cdot 127 \\ m_{32} &= 11 \cdot 19 \cdot 127 \cdot * \\ m_{58} &= 7 \cdot 19 \cdot 127 \cdot * \\ m_{80} &= 5 \cdot 7 \cdot 11 \cdot 127 \cdot * \\ m_{102} &= 7^2 \cdot 11 \cdot 127 \cdot * \\ m_{120} &= 7^2 \cdot 11 \cdot 127 \cdot 419 \cdot 1093 \cdot * \\ m_{138} &= 7 \cdot 11 \cdot 19 \cdot 127 \cdot * \\ m_{160} &= 7 \cdot 11^2 \cdot 19 \cdot 127 \cdot * \end{aligned}$$

where * indicates an integer (not necessarily the same) having no prime factors smaller than 10000.

The primes 7,11,19 and 127 are precisely those obtained in the evaluation of $\sqrt{j(\tau)-1728}$ when $\tau = \frac{-1+\sqrt{-163}}{2}$. For we have, by (3.5.5)

$$j(\tau) = -(2^6.3.5.23.29)^3$$
$$= -262537412640768000$$

and so $$j(\tau)-1728 = -262537412640769728$$
$$= 1610658973256256 \times -163 .$$

Hence $$\sqrt{j(\tau)-1728} = 40133016\sqrt{-163}$$
$$= 2^3.3^3.7.11.19.127\sqrt{-163} .$$

Finally, all these observations on the cubic $x^3 - 6x^2 + 4x - 2 = 0$ carry over to the cubics associated with other imaginary quadratic fields with class-number 1. However, because the absolute values of the discriminants of these fields are less than 163 (see Stark (18)), the phenomena are not nearly so pronounced. Thus, with the discriminants -67 and -43, we have

$$e^{\pi\sqrt{67}} = 14\ 71949\ 52743.99999\ 86624\ 54224\ \ldots$$
$$\text{and } e^{\pi\sqrt{43}} = \quad 8847\ 36743.99977\ 74660\ 34906\ \ldots$$

The cubics satisfied by $f(\sqrt{-67})$ and $f(\sqrt{-43})$ are, respectively,

$$x^3 - 2x^2 - 2x - 2 = 0$$
$$x^3 - 2x^2 \qquad - 2 = 0$$

and one observes the partial quotients 87431 and 29866 respectively early on in the continued fraction expansion of the real root of each cubic.

Below are given the values of Θ and Θ_1 to 200 decimal places, together with a table of the partial quotients a_i and a'_i in their respective continued fraction expansions.

Θ = 5.31862 82177 50185 65910 96801 53318 02246 77219 19808 83690
02602 28091 99584 01958 97457 32187 43665 34591 07487 15400
45589 07647 42444 78645 91488 72327 64878 31165 98454 79445
12414 29908 75700 21982 39534 04098 41477 60189 42443 29911

Θ_1 = 5.31862 82177 50185 65910 96801 53318 02246 77219 19808 77903
25291 02486 16405 49342 75592 94236 99171 57852 32639 72630
30208 06883 94087 98203 79559 02991 44546 73822 67205 03958
81415 44232 97654 81169 19582 87166 38540 80329 70950 41676

i	a_i	a'_i	i	a_i	a'_i	i	a_i	a'_i
0	5	5	36	1	1	72	1	2
1	3	3	37	7	1	73	1	2
2	7	7	38	6	3	74	7	1
3	4	4	39	1	1	75	2	6
4	2	2	40	1	3	76	1	3
5	30	30	41	5	2	77	7	1
6	1	1	42	2	2	78	1	6
7	8	8	43	1	1	79	3	38
8	3	3	44	6	2	80	25	4
9	1	1	45	2	2	81	49405	6
10	1	1	46	2	3	82	1	1
11	1	1	47	1	1	83	1	8
12	9	9	48	2	1	84	3	1
13	2	2	49	1	1	85	1	2
14	2	2	50	1	1	86	1	2
15	1	1	51	3	2	87	4	7
16	3	3	52	1	2	88	1	1
17	22986	22986	53	3	18	89	2	1
18	2	2	54	1	5	90	15	1
19	1	1	55	2	77	91	1	1
20	32	32	56	4	1	92	2	2
21	8	8	57	3	40	93	83	23
22	2	2	58	1	1	94	1	1
23	1	1	59	35657	2	95	162	2
24	8	8	60	1	2	96	2	7
25	55	55	61	17	5	97	1	1
26	1	1	62	2	1	98	1	12
27	5	5	63	15	3	99	1	5
28	2	2	64	1	1	100	2	5
29	28	28	65	1	6	101	2	2
30	1	1	66	2	7	102	1	1
31	5	6	67	1	1	103	53460	1
32	1	110	68	1	6	104	1	1
33	1501790	1	69	5	1	105	6	19
34	1	1	70	3	1	106	4	2
35	2	3	71	2	2	107	3	1

i	a_i	a'_i	i	a_i	a'_i	i	a_i	a'_i
108	4	1	139	48120	7	170	1	10
109	13	4	140	1	2	171	1	1
110	5	1	141	2	2	172	1	4
111	15	11	142	17	2	173	2	2
112	6	2	143	2	4	174	2	4
113	1	4	144	1	3	175	2	1
114	4	1	145	2	2	176	2	19
115	1	6	146	1	1	177	2	16
116	4	1	147	4	4	178	17	1
117	1	1	148	2	6	179	4	1
118	1	1	149	3	1	180	9	4
119	2	2	150	1	1	181	5	1
120	1	1	151	2	1	182	1	14
121	16467250	1	152	23	1	183	7	500
122	1	1	153	3	24	184	11	4
123	3	5	154	2	1	185	1	2
124	1	4	155	1	1	186	2	3
125	7	5	156	1	1	187	9	24
126	2	3	157	1	14	188	1	1
127	6	6	158	2	1	189	14	4
128	1	4	159	1	1	190	4	1
129	95	2	160	27	9	191	6	1
130	20	2	161	325927	1	192	1	1
131	1	4	162	1	1	193	22	2
132	2	1	163	60	5	194	11	3
133	1	1	164	1	1	195	1	1
134	6	1	165	87	3	196	1	3
135	1	1	166	1	1	197	1	16
136	1	1	167	2	2	198	1	2
137	8	9	168	1	8	199	4	1
138	1	1	169	5	1	200	1	1

3.6. Some determinants connected with the zeros of Eisenstein series.

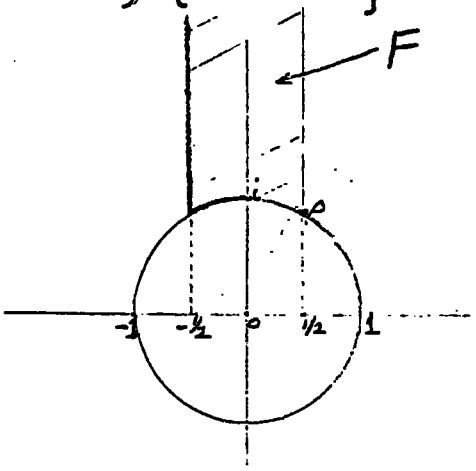
This section deals with certain computations connected with Eisenstein series. The full details can be found in R.A.Rankin's paper "The zeros of Eisenstein series" - Rankin (16).

The Eisenstein series are defined for even $k \geqslant 4$ by

$$E_k(z) = \tfrac{1}{2}\sum_{(c,d)=1} (cz + d)^{-k} \qquad (\Im z > 0) \quad .$$

Let M_k denote the space of entire modular forms of dimension $-k$ for the modular group $\Gamma(1)$:- see 2.2 . The fundamental region for $\Gamma(1)$, denoted by F, will be taken as $A_1 \cup A_2 \cup A_3$ where

$$A_1 = \left\{ z : |z| \geqslant 1, \ -\tfrac{1}{2} \leqslant Rz \leqslant 0 \right\}$$
$$A_2 = \left\{ z : |z| > 1, \ 0 \leqslant Rz < \tfrac{1}{2} \right\}$$
$$A_3 = \left\{ z : z = \infty \right\}$$



It can be shown that any member of M_k has $\frac{k}{12}$ zeros in F, if zeros at z=i are counted with multiplicity $\frac{1}{2}$, those at $z=\rho$ with multiplicity $\frac{1}{3}$, and zeros elsewhere with multiplicity 1. Wohlfahrt (21) showed that for even k satisfying $4 \leqslant k \leqslant 26$, all the zeros of $E_k(z)$ in F lie on the circle $|z|=1$. Rankin, in his paper (16) discusses the conjecture that this property of the zeros of $E_k(z)$ in F holds for all $k \geqslant 4$, and some evidence for the truth of this conjecture is given in this paper. Rankin proves that

(1) if $k \equiv 2 \pmod 4$,

then $E_k(z) \neq 0$ for $y = \Im z > 1$

(i.e. the conjecture holds for this case)

(2) if $k \equiv 0 \pmod 4$ and $y = \sqrt{z}$

then $E_k(z) \not\equiv 0$ providing that $y > 1 + \frac{1}{2\pi} \log C_k$

where $C_k = \frac{\alpha_k}{\sqrt{\alpha_{k-2}\alpha_{k+2}}}$, and $\alpha_k = \frac{(2\pi)^k}{(k-1)!\zeta(k)}$

or, alternatively, $\alpha_k = \frac{B_{2k}}{2k}$ where B_n is the n-th.

Bernoulli number in the even-suffix notation ($B_2 = \frac{1}{6}$).

In an attempt to disprove the conjecture for sufficiently large k, Rankin was led to consider certain determinants, defined in the following way.

Let $x = e^{2\pi i z}$. The modular invariant j(z), see 2.2., has the Fourier expansion:

$$j(z) = x^{-1} \sum_{n=0}^{\infty} a_n x^n \text{ , with } a_1 \text{ taken as } 744 \text{ .}$$

Define $a_n^{(\nu)}$ by

$$j^{\nu}(z) = x^{-\nu} \sum_{n=0}^{\infty} a_n^{(\nu)} x^n \; ; \quad \nu \geqslant 0$$

and let

$$g_\nu = a_\nu^{(\nu)} - 24 \sum_{m=1}^{\nu} a_{\nu-m}^{(\nu)} \sigma(m)$$

where

$$\sigma(m) = \sum_{d/m} d$$

Finally, for $n \geqslant 0$, let

$$\Delta_n = \begin{vmatrix} g_0 & g_1 & g_2 & \cdots\cdots & g_n \\ g_1 & g_2 & g_3 & \cdots\cdots & g_{n+1} \\ \cdot\cdot & \cdot\cdot & \cdot\cdot & \cdots\cdots & \cdot\cdot \\ \cdot\cdot & \cdot\cdot & \cdot\cdot & \cdots\cdots & \cdot\cdot \\ g_n & g_{n+1} & g_{n+2} & \cdots\cdots & g_{2n} \end{vmatrix}$$

We find that :

$$\Delta_0 = |1| = 1$$

$$\Delta_1 = \begin{vmatrix} 1 & 720 \\ 720 & 911520 \end{vmatrix} = 393120$$

$$\Delta_2 = \begin{vmatrix} 1 & 720 & 911520 \\ 720 & 911520 & 1301011200 \\ 911520 & 1301011200 & 1958042030400 \end{vmatrix}$$

$$= 27454623356160000 .$$

By comparing the determinants Δ_n with a sum of products

$$\prod_{\substack{i<j \\ 1\leq j\leq n}} (\alpha_i - \alpha_j)^2$$

where $\alpha_0, \alpha_1, \ldots, \alpha_n$ assume all sets of n+1 values $j(\zeta)$ taken from the zeros ζ of E_k, and by using the fact that j is real on the boundary of the fundamental region F, Rankin proves that if, for any n, $\Delta_n < 0$ then there exists a k_0 such that, for all $k \geq k_0$, E_k does not have all its zeros in F situated on the boundary of F ; i.e. E_k certainly has zeros with $|z| \neq 1$.

The author computed Δ_n , n=0,1,2,...,13, using the multi-length package described in 3.2 . These determinants were all positive and monotonically increasing in value. One peculiarity of these determinants was that each of them turned out to be products of powers of small primes. This was not immediately obvious from the definition of the elements g_n of the determinants. Each g_n had various powers of 2 and 3 in their factorisation but otherwise were not highly composite. If they had been, this would have added to the "compositeness" of the Δ_n. A table of the factorisation of $\Delta_1, \Delta_2, \ldots, \Delta_{13}$ is given below. It lists for each prime p dividing Δ_n the power of p in the factorisation ; a blank entry denotes zero.

p	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7	Δ_8	Δ_9	Δ_{10}	Δ_{11}	Δ_{12}	Δ_{13}
2	5	11	18	25	32	40	49	57	64	72	81	90	99
3	3	6	10	15	18	22	27	32	37	42	47	53	60
5	1	4	5	7	9	11	14	15	17	19	23	28	29
7	1	2	3	5	7	9	11	13	15	17	19	21	23
11		1	2	3	5	5	6	7	8	11	13	14	16
13	1	2	3	4	5	7	7	8	9	10	11	12	13
17		1	2	3	4	5	7	9	9	9	10	11	12
19		1	2	3	4	5	6	7	9	9	10	11	12
23			1	2	3	4	5	6	7	9	11	11	11
29			1	2	3	4	5	6	7	8	9	11	13
31			1	2	3	4	5	6	7	8	9	10	11
37			1	2	3	4	5	6	7	8	9	10	11
41				1	2	3	4	5	6	7	8	9	10
43				1	2	3	4	5	6	7	8	9	10
47					1	2	3	4	5	6	7	8	9
53					1	2	3	4	5	6	7	8	9
59						1	2	3	4	5	6	7	8
61					1	2	3	4	5	6	7	8	9
67						1	2	3	4	5	6	7	8
71							1	2	3	4	5	6	7
73						1	2	3	4	5	6	7	8
79							1	2	3	4	5	6	7
83								1	2	3	4	5	6
89								1	2	3	4	5	6
97								1	2	3	4	5	6
101									1	2	3	4	5
103									1	2	3	4	5
107										1	2	3	4
109									1	2	3	4	5
113										1	2	3	4
127											1	2	3
131												1	2
137												1	2
139												1	2
149													1
151													1
157													1

One fact immediately noticeable from the above table is that, for $n=1,2,\ldots,13$, Δ_{n-1} divides Δ_n. Let $\Delta'_n = \frac{\Delta_n}{\Delta_{n-1}}$, $n \geqslant 1$.

Thus $\Delta'_1 = 393120$

$\Delta'_2 = 69837768000$.

The factorisation of Δ'_n, $n=1,2,\ldots,13$ is given in the first 13 columns of the table below.

P \ Δ'	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	5	6	7	7	7	8	9	8	7	8	9	9	9	10	11	9	7	8	9	9	9	10	11	10	9
3	3	3	4	5	3	4	5	5	5	5	5	6	7	3	4	5	5	5	5	5	6	7	5	6	7
5	1	3	1	2	2	2	3	1	2	2	4	5	1	2	3	3	4	2	3	3	3	4	2	4	3
7	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	1
11	0	1	1	1	2	0	1	1	1	3	2	1	2	2	2	3	1	2	2	2	3	2	1	2	2
13	1	1	1	1	1	2	0	1	1	1	1	1	1	2	2	2	2	2	3	1	2	2	2	2	2
17		1	1	1	1	1	2	2	0	0	1	1	1	1	1	2	1	0	1	1	1	1	1	3	3
19		1	1	1	1	1	1	1	2	0	1	1	1	1	1	1	1	2	1	0	1	1	1	1	1
23			1	1	1	1	1	1	1	2	2	0	0	1	1	1	1	1	1	1	2	2	1	0	0
29			1	1	1	1	1	1	1	1	1	2	2	2	0	0	0	1	1	1	1	1	1	1	1
31			1	1	1	1	1	1	1	1	1	1	1	2	2	0	0	1	1	1	1	1	1	1	1
37			1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	0	0	0	1	1	1	1
41				1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	0	0	0	0	1
43				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	0	0	0	1
47					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	0	0
53					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2
59					0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
61					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
67						1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
71						0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
73						1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
79							1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
83								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
89								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
97								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
101									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
103									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
107									0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
109									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
113										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
127											1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
131												1	1	1	1	1	1	1	1	1	1	1	1	1	1
137												1	1	1	1	1	1	1	1	1	1	1	1	1	1
139												1	1	1	1	1	1	1	1	1	1	1	1	1	1
149													1	1	1	1	1	1	1	1	1	1	1	1	1
151													1	1	1	1	1	1	1	1	1	1	1	1	1
157													1	1	1	1	1	1	1	1	1	1	1	1	1
163														1	1	1	1	1	1	1	1	1	1	1	1
167															1	1	1	1	1	1	1	1	1	1	1
173															1	1	1	1	1	1	1	1	1	1	1
179															0	1	1	1	1	1	1	1	1	1	1
181															1	1	1	1	1	1	1	1	1	1	1
191																0	1	1	1	1	1	1	1	1	1
193																1	1	1	1	1	1	1	1	1	1
197																	1	1	1	1	1	1	1	1	1
199																	1	1	1	1	1	1	1	1	1

(contd. over)

p \ Δ'	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
211																		1	1	1	1	1	1	1	1
223																			1	1	1	1	1	1	1
227																			0	1	1	1	1	1	1
229																			1	1	1	1	1	1	1
233																				1	1	1	1	1	1
239																				0	1	1	1	1	1
241																				1	1	1	1	1	1
251																						1	1	1	1
257																						1	1	1	1
263																							1	1	1
269																							1	1	1
271																							1	1	1
277																							1	1	1
281																								1	1
283																								1	1
293																									1

To obtain more data, the Chinese Remainder Theorem was used to compute $\Delta'_{14}, \ldots, \Delta'_{25}$, instead of computing the Δ'_n directly by the multi-length package which was already proving expensive in computing time. A large prime, p, was chosen and all computation performed modulo p. In this particular case, p was of the order of 10^8. The Fourier coefficients in the expansion of j were computed (mod p) using a subroutine written for this purpose, namely KLEINJ (see 4.5.). The determinant Δ_n was then obtained with its elements $g_0, \ldots, g_{2n}$ computed mod p. To obtain a list of the Δ'_n, it was necessary to set up Δ_N, where N was the largest n for which Δ'_n was to be computed - in this case N was 25. Reducing Δ_N to a triangular matrix by Gauss elimination then gave the Δ'_n as the diagonal elements of this matrix. These elements were recorded. The whole process was then repeated sufficiently many times modulo different primes and the Chinese Remainder Theorem then applied to obtain the Δ'_n modulo the product of these primes. Just how many moduli need to be used in an application of

the Chinese Remainder Theorem depends on how large the answer is; in this particular application this was not known, though an estimate as to how large Δ'_n might be expected to be was obtained by a crude extrapolation of the values of $\Delta'_1, \ldots, \Delta'_{13}$. A very large answer, near to the product of the moduli, would have meant that either too few moduli had been used in the computation or the result was a negative number or a fraction or possibly both. This latter situation would have spoiled the conjecture that Δ_n was divisible by Δ_{n-1} for $n \geqslant 1$. The results, as it turned out, clearly suggested that the Δ'_n were positive integers and the factorisation of these numbers for $1 \leqslant n \leqslant 25$ is given in the previous table, the first 13 columns checking with the previous calculation of these numbers by the multi-length package.

Let now $\Delta''_n = \Delta'_n / \Delta'_{n-1}$, $n \geqslant 2$. Thus $\Delta''_2 = 2.5^2.11.17.19$, $\Delta''_3 = 2.3.5^{-2}.23.29.31.37$, and $\Delta''_4 = 3.5.7.41.43$. From a table of $\Delta''_2, \ldots, \Delta''_{25}$ the author conjectured that, for $n \geqslant 2$,

$$\Delta''_n = \frac{36(12n+1)(12n-5)(12n-7)(12n-13)}{n(n-1)(2n-1)^2} . \qquad (3.6.1)$$

If this conjecture is true, then obviously $\Delta_n > 0$ for $n \geqslant 0$ and no light is shed on Rankin's original problem, namely, to show that, for sufficiently large k, E_k has all its zeros in F on the circle $|z| = 1$.

In conclusion, it should be noted that a similar situation arises with the expansion of certain Eisenstein series and powers of these in terms of j. We give three examples.

(1) Write $E_4(z)$ as $1 + 240\sum_{n=1}^{\infty}\sigma_3(n)x^n$,

where $\sigma_3(n) = \sum_{d|n} d^3$ and $x = e^{2\pi i z}$. We take

$$j(z) = x^{-1} + 744 + 196884x + \cdots .$$

If $\sqrt[4]{E_4(z)}$ is expanded as a power series in j^{-1}, i.e. if

$$\sqrt[4]{E_4} = 1 + \sum_{n=1}^{\infty} a_n j^{-n} \tag{3.6.2}$$

then the a_n enjoy a property similar to that found for the Δ''_n in (3.6.1). From the fact that $\sqrt[4]{E_4}$ is a hypergeometric function of $12^3/j$, it can be shown that if, for $n \geq 1$, $b_n = \frac{a_n}{a_{n-1}}$ with $a_0 = 1$, then

$$b_n = \frac{12(12n - 7)(12n - 11)}{n^2} \tag{3.6.3}$$

(2) By a change of variable, it may also be shown that if

$$\sqrt[4]{E_4}\,(1 - 1728j^{-1})^{-1/2} = 1 + \sum_{n=1}^{\infty} a'_n j^{-n}$$

and $b'_n = a'_n/a'_{n-1}$, then

$$b'_n = \frac{12(12n - 1)(12n - 5)}{n^2} \tag{3.6.4}$$

(3) Finally, if

$$\sqrt{E_4} = 1 + \sum_{n=1}^{\infty} a''_n j^{-n}$$

and $b''_n = a''_n/a''_{n-1}$, then

$$b''_n = \frac{24(6n - 1)(6n - 5)(2n - 1)}{n^3} \tag{3.6.5}$$

which follows from (3.6.3) by Clausen's formula.

CHAPTER 4.

4.1. Subroutines and functions.

The concept of a library of subroutines has been introduced in 1.4. In this chapter, we describe certain properties of a number-theoretic subroutine library and give computer-printed Fortran listings of the subroutines. Some of these subroutines are written in the language ASP. As the statements in this language are virtually basic machine-code, a listing of an ASP subroutine is not given, since it is likely to be of interest only to those programmers acquainted with Atlas machine-code. The version of Fortran used is that of the Hartran System available on the Atlas 1 Chilton installation (see ICL(8)).

Subroutines may be divided into two classes, namely subroutines proper and functions. Basically, the only distinction is that, while a subroutine or a function is entered by a CALL statement from the calling programme, a function may also be entered by its name occurring in the calling programme as though it were a variable (with its arguments, if any). In this case the value of the function is that of the accumulator on exit from the function.

Each listing of a subroutine or function is preceded by "comment" cards giving the purpose of the routine, the arguments (if any), restrictions (if any) on use, subroutines called (if any), b-registers used and, if necessary, any further comments. Where variable-precision arithmetic subroutines are called, these routines will be denoted collectively by ML.

4.2. Testing of subroutines.

The restrictions on the use of certain of the subroutines arise either from the way in which the subroutine was coded or from restrictions on the use of certain machine orders. Failure to observe the first type of restriction is either trapped by the programme itself, and a suitable message printed, or is liable to cause overwriting with possibly disastrous results. Failure to observe the second type of restriction will almost certainly produce incorrect results.

The flow-diagram of a subroutine may be quite a complicated structure. Some sections of the code may only rarely be executed; this fact will usually make it difficult to thoroughly test a given subroutine. Continual use is, in the end, perhaps the best method of testing.

4.3. "Overwriting" functions.

The appearance of a subroutine or function name in a Hartran System Fortran programme causes the following code ("calling sequence") to be generated by the compiler:

(1) record the return address, R say, in a b-register;

(2) go to the subroutine or function (on exit return to R);

(3) give information for error-tracing;

(4) generate a list of addresses of the arguments.

We shall call a subroutine or function "overwriting" if, on its first execution, it plants the relevant machine-code directly in the calling sequence and

overwrites the code in (1) in such a way that the code in (2) is bypassed. With this device, the subroutine or function is never accessed again on subsequent calls (unless there are calls for it elsewhere in the calling programme).

The construction of an "overwriting" routine is only possible if the relevant code of the routine consists of approximately as many instructions as there are relevant arguments of the routine (this is due to the way in which the list (4) is generated). All the "overwriting" routines in the number-theoretic library are in fact functions, leaving their answers in the accumulator. Furthermore, some of these functions require dummy arguments which must appear in the argument list and may assume any value, conveniently zero.

Finally, if a programme uses these "overwriting" functions, incorrect information may be output by the Hartran System error-tracing routine. This is due to the fact that the information in (3) is overwritten in the first execution run of some of these functions.

4.4. The Fortran used in the subroutine library.

The version of Fortran that has been used in the subroutine library given in 4.5. differs slightly from that issued by the American Standards Association (Communications of the A.C.M., vol. 7, No. 10, Oct. 1964). In particular, the following deviations have been made:-

(1) Arithmetic IF statement.

```
IF (I) 1,,
```

is equivalent to

```
IF (I) 1,0,0
```

where a 0 label indicates the next statement. Similarly

IF (I) ,1, is equivalent to IF (I) 0,1,0

(2) Logical IF statement.

```
IF (L) 1,
next statement
```

Where L is a logical expression, is equivalent to

```
IF (L) GO TO 1
```

and IF (L) ,1 is equivalent to

```
IF (L) GO TO 2
GO TO 1
2 next statement
```

(3) DO - loop.

On satisfying a do-loop, the value of the index is "correct", i.e. on satisfying

```
DO 1 I = N1,N2,N3
 .
 .
 .
1 statement
```

the value of I is N1+k.N3 where k is the largest integer such that N1+k.N3 $\leq$ N2.

(4) FOR - loop

```
FOR I = N1,N2,N3
 .
 .
 .
```

REPEAT

is identical to

```
DO 1 I = N1,N2,N3
 .
 .
 .
1 statement
```

except that if N2<N1, the for-loop will not be executed and the do-loop will be executed once (assuming throughout that N3 is positive).

(5) Array storage.

Arrays or variables coded next to each other are stored continguously. Thus the statement

```
INTEGER A(100),B(50),C
```

implies that A(101) and B(1) share the same store location, as do A(100) and B(0), and B(51) and C. This fact is assumed, for example, in the subroutine DC.

(6) Print statement.

```
PRINT 100, Imput/output list
```

is equivalent to

```
WRITE (N,100) Imput/output list
```

where N is an integer constant or variable specifying the lineprinter (on Atlas, N specifies an output stream destined for the lineprinter).

(7) Truncation statement.

```
TRUNCATION INTF
```

indicates that the statement I = X, where I is integer and X is real, has the effect of setting I to contain the integer part of X.

(8) Comment statement.

```
statement      π comment
```

is equivalent to

```
statement
C     comment
```

Some Hartran system routines are used in certain subroutines. They are:-

IOZ (called from READML)

CALL IOZ(N), where N is an integer constant or variable will cause at least N digits to be output by an Iw descriptor in a format specification. The subroutine is thus used to output leading zeros rather than suppress them. wZ appearing in a format specification is equivalent to a call of IOZ with N = w.

IABSF (called from FACTOR)

N = IABSF (I) sets N to contain I ; N and I are integer variables.

PRIFAC (called from FACTOR)

CALL PRIFAC (I1,I2,I3,I4) where I1 is an integer array and I2,I3 and I4 are integer variables, will insert into the output buffer I1(1), I1(2), ..., I1(I2), in such a way that CALL OUTREC will produce a line of output containing

$$I1_1(I1_2).I1_3(I1_4)\ldots I1_{I2-1}(I1_{I2}),$$

followed by an asterisk if I4 is non-zero. (Here $I1_k$ = I1(k)). If I1(2n) is 1 then it is omitted. If the output buffer is filled completely by executing PRIFAC then I3 will be greater than 160.

OUTREC (called from FACTOR)

CALL OUTREC will print the output buffer. Thus filling the output buffer and calling OUTREC is equivalent to a PRINT statement.

INPSEL (called from MLJ)

CALL INPSEL (N) where N is an integer constant or variable will select imput stream N (in Atlas terminology). In MLJ, this routine is used in conjunction with the multi-length routine READ which is given as an argument to the routine MLO (or MLT). READ will input multi-length

variables from the currently selected input stream.

SINF (called from MLJ)

X = SINF(Y) will set X to contain sin Y.

The subroutine listings given in 4.5 are in the following order:-

(i) "single-length" subroutines, alphabetically

(ii) "multi-length" subroutines, (i.e. those subroutines that call the multi-length routines) alphabetically.

REFERENCES.

1. Atkin, A.O.L. Congruences for modular forms, in Computers in Mathematical Research. Editors Churchhouse and Herz, North-Holland (1968).

2. Atkin, A.O.L. and Swinnerton-Dyer, H.P.F. Modular forms on non-congruence subgroups, Proceedings of the Symposium in Combinatorics, American Mathematical Society, Los Angeles (1968), to appear.

3. Baker, A. Linear forms in the logarithms of algebraic numbers, to appear.

4. Baker, A. and Davenport, H. The equations $3x^2-2=y^2$ and $8x^2-7=z^2$, to appear.

5. Cassels, J.W.S. Diophantine equations with special reference to elliptic curves, Journal of the London Mathematical Society, Vol. 41, (1966), 193-291.

6. Churchhouse, R.F. and Muir, S.T.E. Continued fractions, algebraic numbers and modular invariants, to appear.

7. Delone, B.N. and Faddeev, D.K. The theory of irrationalities of the third degree, Vol. 10, Translations of Mathematical Monographs. American Mathematical Society (1964).

8. ICL (International Computers Limited). A primer of Fortran programming for use on Atlas and Orion computers. (CS 390).

9. ICL (International Computers Limited). Interasp - an intermediate Atlas Symbolic programming language. (AERE R4285).

10. Khinchin, A. Continued Fractions, Chapter III. Noordhoff Ltd. (1963).

11. Klein, F. and Fricke, R. Vorlesungen über die theorie der elliptischen modulfunktionen, Vol. I. Teubner, Stuttgart.

12. Kusmin, R.O. On a problem of Gauss. Reports of the Academy of Sciences (A), 1928, 375-380.

13. Lehmer, D.H. Machines and pure mathematics, in Computers in Mathematical Research. Editors Churchhouse and Herz, North-Holland (1968).

14. Lehmer, D.H. Properties of the coefficients of the modular invariant $J(\tau)$, American Journal of Mathematics 64 (1942), 488-502.

15. Lévy, P. Théorie de l'addition des variables aléatoires, Paris (1937).

16. Rankin, R.A. The zeros of Eisenstein series, to appear.

17. Siegel, C.L. The integer solutions of the equation $y^2=ax^n+bx^{n-1}+\ldots+k$, Journal of the London Mathematical Society, Vol. 1, (1926), 66-68.

18. Stark, H.M. There is no tenth complex quadratic field with class-number 1, Proceedings of the National Academy of Sciences, Vol. 57, (1967), 216-221.

19. Swinnerton-Dyer, H.P.F. An application of computing to class field theory, in Algebraic Number Theory. Editors Cassels and Fröhlich, Academic Press (1967).

20. Weber, H. Lehrbuch der Algebra. Vol.III. Chelsea Publishing Company.

21. Wohlfahrt, K. Über die Nullstellen einiger Eisensteinreihen, Math. Nachr. 26, (1964), 381-383.

22. Wrench, J.W. and Shanks, D. Questions concerning Khintchine's constant and the efficient computation of regular continued fractions, Mathematics of Computation, Vol. 20 (1966), 444-448.

23. Van Lint, J.H. On a set of Diophantine equations. Report 68-WSK-03 of the Technological University Eindhoven, (1968).

24. X. See Siegel (17).

C 4.5. THE SUBROUTINE LIBRARY.

C ABBREVIATIONS

C THE RESTRICTION $1 \leq N \leq 2^{35}-1$ IS DENOTED BY R*(N)

C THE RESTRICTION $0 \leq N \leq 2^{35}-1$ IS DENOTED BY R0(N)

C THE RESTRICTION $0 \leq I,J \leq 2^{35}-1$ IS DENOTED BY RMOD(M,I,J)
C $1 \leq M \leq 2^{35}-1$
C $(I*J)/M < 2^{39}$.

C THE RESTRICTION $0 \leq I \leq 2^{35}-1$ IS DENOTED BY RMOD(M,I)
C $1 \leq M \leq 2^{35}-1$

C THE RESTRICTION $M\sqrt{2N/3} < F$ IS DENOTED BY REUL(M,N)

C THE RESTRICTION $M\sqrt{2N/(3NN)} < F$ IS DENOTED BY REULQ(M,N,NN)

C THE RESTRICTION 'MUST NOT BE USED IN A PRINT STMT.' IS DENOTED BY RPRINT

C THE RESTRICTION 'MUST NOT BE USED IN AN IF STMT.' IS DENOTED BY RIF

C $F = 2^{36}-1$

C $$F(X) = \prod_{M=1}^{\infty} (1 - X^M)$$

C ML = MULTI-LENGTH = VARIABLE PRECISION.

```
      SUBROUTINE BQF(V,FA,FB,FC,FD,N,Z,IPRINT,IUNLD)

C  PURPOSE,  BINARY QUADRATIC FORMS.
C  ARGUMENTS.  FA,FB,FC,FD INTEGER ARRAYS. A REDUCED SET OF BINARY QUADRATIC
C              FORMS
C                  FA(I)   ± FB(I)     FC(I),
C              I = 1,N IS CALCULATED FOR DISCRIMINANT =V.  ON RETURN FD MAY CONTAIN
C              Z = CLASS NUMBER OF DISCRIMINANT -V.  IF IPRINT NEQ 0,  FORMS ARE
C              PRINTED WITH ** AFTER THEM IF HCF OF FA(I),FB(I),FC(I) NEQ 1,
C              AND WITH * AFTER THEM IF FB(I) IS POSITIVE(AND NOT NEGATIVE).
C  RESTRICTIONS.  3 ≤ V < 1,000,000.  V ≡ 3( MOD 4)
C  CALLS  IZQ,IZR
C  B-REGS.  0 - 20
C        NEEDS PRIMES AT BLOCK 1 ON TAPE 2,WHICH IS UNLOADED IF IUNLD ≠ 0.

      INTEGER A,B,C,D,G,H,P,R,V,Z,FDD,D4
      INTEGER X(7),Q(7),E0,E(7),F(200),FF(200)
      TEXT STAR,STARS(500)
      INTEGER FA(1),FB(1),FC(1),FD(1)
101   FORMAT(5HOD = ,I6,6X,4HH = ,I4)
102   FORMAT(5(1X,3I6,A5))
      IF(III) 3,,3
      REWIND 2
      READ TAPE 2,F
      DO 4 P = 1,200
4     FF(P) = F(P)*F(P)
      IF(IUNLD) ,3,
      CALL UNLOAD(2)
3     D = V
      IF(D) 2,2,
      LL = IZR(4,D-3)
      IF(LL) ,1,
      PRINT 8
8     FORMAT(8H BQF,    ,5X,14H-D NEQ 3 MOD 4)
      RETURN
1     D4 = 4*D
      R,JX,N = 0
      P = 2
5     LL = IZR(F(P),D)
      IF(LL) 7,,7
      LL = IZR(FF(P),D)
      IF(LL) 7,,7
      R = R + 1
      Q(R) = F(P)
7     P = P + 1
      IF(3*FF(P) - D) 5,5,
      IF(R) 43,,43
      JX = 0
      GO TO 44
43    JX = 1
```

```
44    FOR JV = R+1,7
      Q(JV) = DA
      REPEAT
9     Z = 0
      B = 1
10    H = (D + B*B)/4
      FOR I = 1,7
      E(I),X(I) = 0
      REPEAT
      K = 0
      E(0),P = 1
12    LL = IZR(F(P),H)
      IF(LL) ,14,
      P = P + 1
      IF(3*FF(P) - D) 12,12,15
14    IF(F(P) - E(K)) ,45,
      K = K + 1
45    E(K) = F(P)
      X(K) = X(K) + 1
      H = IZQ(0,0,0,F(P),H)
      GO TO 12
15    IF(K) 46,,46
      JX = JX + 1
46    FOR L = K+1,7
      X(L) = 0
      E(L) = 1
      REPEAT
      FOR JA = 0,X(1)
      FOR JB = 0,X(2)
      FOR JC = 0,X(3)
      FOR JD = 0,X(4)
      FOR JE = 0,X(5)
      FOR JF = 0,X(6)
      FOR JG = 0,X(7)
      A = 1
      IF(JA) ,35,
      FOR JH = 1,JA
      A = A*E(1)
      REPEAT
35    IF(JB) ,36,
      FOR JH = 1,JB
      A = A*E(2)
      REPEAT
36    IF(JC) ,37,
      FOR JH = 1,JC
      A = A*E(3)
      REPEAT
37    IF(JD) ,38,
      FOR JH = 1,JD
      A = A*E(4)
      REPEAT
```

```
38    IF(JE) ,39,
      FOR JH = 1,JE
      A = A*E(5)
      REPEAT
39    IF(JF) ,40,
      FOR JH = 1,JF
      A = A*E(6)
      REPEAT
40    IF(JG) ,16,
      FOR JH = 1,JG
      A = A*E(7)
      REPEAT
16    IF(B - A) ,,34
      C = (D + B*B)/(4*A)
      IF(A - C) 20,,20
      IF(JY) ,31,
      JZ = 0
      GO TO 25
20    IF(C - A) 34,,
      IF(A - B) 23,,23
      IF(JY) ,31,
      JZ = 0
      GO TO 25
23    IF(JY) ,32,
      JZ = 1
25    G = 1
26    LL = IZR(Q(G),B)
      IF(LL) 28,,28
      LL = IZR(Q(G),A)
      IF(LL) 28,29,28
28    G = G + 1
      IF(Q(G) - B) 26,26,30
29    LL = IZR(Q(G),C)
      IF(LL) 28,33,28
30    IF(JZ) 32,31,32
31    Z = Z + 1
      STAR = 3H *
      FDD = 1
      GO TO 50
32    Z = Z + 2
      STAR = 3H
      FDD = 0
      GO TO 50
33    STAR = 3H **
      FDD = -1
50    N = N + 1
      FA(N) = A
      FB(N) = B
      FC(N) = C
      FD(N) = FDD
      STARS(N) = STAR
```

```
34    REPEAT
      REPEAT
      REPEAT
      REPEAT
      REPEAT
      REPEAT
      REPEAT
      REPEAT
      B = B * 2.
      IF(3*B*B1 - D) 10,10,
      IF(IPRINT) ,2,
      PRINT 101,-D,Z
      NN = (N+4)/5
      DO 51 MM = 1,NN
51    PRINT 102,(FA(M),FB(M),FC(M),STARS(M),M = MM,N,NN)
2     RETURN
      END
```

```
      SUBROUTINE COEFF(X,M,Y,N,K,Z,MM)

C  PURPOSE,  Z = COEFFICIENT OF A**K  (MOD MM)  IN X*Y  WHERE
C          X = A**M(...,    Y = A**N(...
C  ARGUMENTS,  X,Y INTEGER ARRAYS.  MM IS MODULUS.
C  RESTRICTIONS,  RMOD(MM,X(I),Y(K-I)),I = M,K-N.
C                 (K+1-N-M)*MM < F
C  CALLS  IZMR,IZR
C  B-REGS,  0 - 11
C        ELEMENTS OF X ACCESSED :   M  TO  K-N
C        ELEMENTS OF Y ACCESSED :   K-M  TO  N

      INTEGER Z,W,X(1),Y(1)
      W = 0
      FOR I = M,K-N
      W = W + IZMR(MM,0,X(I),Y(K-I))
      REPEAT
      Z = IZR(MM,W)
      RETURN
      END
```

```
      SUBROUTINE CONFRA(XX,I,J,K,A,LOUT,NTEST)

C  PURPOSE,  COMPUTES CONTINUED FRACTION EXPANSION OF XX.
C  ARGUMENTS,  I,J,K  INTEGER ARRAYS,  ON RETURN I,J,K CONTAIN PARTIAL QUOTIENT
C              NUMERATOR AND DENOMINATOR OF CONVERGENTS TO XX. LOUT SET TO
C              NUMBER OF PARTIAL QUOTIENTS COMPUTED.  COMPUTATION STOPS WHEN
C              NUMERATOR + DENOMINATOR >=10**9.  A IS REAL ARRAY CONTAING
C              NUM./DEN. = XX.  PRINTS CONTINUED FRACTION IF NTEST NEQ 0
C  B=REGS,  0 - 10
C        ML VERSION IS MLCVS

      DIMENSION I(1),J(1),K(1),A(1)
      TRUNCATION INTF
      J(1),K(2) = 0
      J(2),K(1) = 1
      L = 3
      X = ABSF(XX)
      Y = X
5     I(L) = Y
      J(L) = I(L)*J(L-1)+J(L-2)
      K(L) = I(L)*K(L-1)+K(L-2)
      B = J(L)
      C = K(L)
      A(L) = B/C - X
      IF(J(L) + K(L) - 1000000000) 4,2,2
4     Z = Y - I(L)
      IF(Z - 0.0000000001) 2,3,3
2     LOUT = L
      GO TO 100
3     Y = 1/Z
      L = L + 1
      GO TO 5
100   IF(NTEST) ,102,
      PRINT 10,XX
10    FORMAT(15H CONVERGENTS OF,F17.10)
      DO 103 L = 3,LOUT
103   PRINT 104,L,I(L),J(L),K(L),A(L)
104   FORMAT(1X,4I12,F17.10)
102   RETURN
      END
```

```
      SUBROUTINE DOMODM(IX,NN,ID,MM)

C  PURPOSE.  TESTS IF DISCRIMINANT DIVISIBLE BY MM.
C  ARGUMENTS,  IX INTEGER ARRAY.  ID SET TO 0 IFF DISCRIMINANT OF IX(1) X**NN +
C              ... + IX(NN+1) = 0 (MOD MM).  OTHERWISE ID NEQ 0 (FAULT GIVES ID
C               =-1)
C  RESTRICTIONS,  NN < 500.  R+(MM)
C  CALLS  IJMODK,IZR
C  B-REGS,  0 - 11

      INTEGER X0,X(500),Y0,Y(500),IX(1)
      N = NN
      M = MM
      Y(N) = 0
      DO 1 I = N+1,1,-1
      IF(IX(I)) ,8,8
      X(N+1-I) = M-IZR(M,-IX(I))
      GO TO 1
 8    X(N+1-I) = IZR(M,IX(I))
 1    CONTINUE
      DO 2 I = 0,N-1
 2    Y(I) = IJMODK(M,0,I+1,X(I+1))
      LX = N
      LY = N-1
      DO 5 K = 1,N+10
 14   LXY = LX - LY
      IF(LXY) 6,,
      FOR I = 0,LXY-1
      X(I) = IJMODK(M,0,Y(LY),X(I))
      REPEAT
      DO 3 I = LXY,LX-1
      IA = IJMODK(M,0,X(I),Y(LY)) - IJMODK(M,0,X(LX),Y(I-LXY)) + M
 3    X(I) = IZR(M,IA)
      LX = LX - 1
      IF(LX) 11,11,
      IF(X(LX)) 6,,6
      LLX,LX = LX-1
      IF(LX) 11,11,
      DO 9 I = LLX,0,-1
      IF(X(I)) 6,,6
      LX = LX - 1
      IF(LX) 11,11,
 9    CONTINUE
 11   ID = X(0)
      RETURN
 6    LXY = LY - LX
      IF(LXY) 14,,
      FOR I = 0,LXY-1
      Y(I) = IJMODK(M,0,X(LX),Y(I))
      REPEAT
```

```
      DO 4 I = LXY,LY-1
      IA = IJMODK(M,0,Y(I),X(LX)) - IJMODK(M,0,Y(LY),X(I-LXY)) + M
4     Y(I) = IZR(M,IA)
      LY = LY - 1
      IF(LY) 12,12,
      IF(Y(LY)) 5,,5
      LLX,LY = LY-1
      IF(LY) 12,12,
      DO 10I = LLX,0,-1
      IF(Y(I)) 5,,5
      LX = LY - 1
      IF(LY) 12,12,
10    CONTINUE
12    ID = Y(0)
      RETURN
5     CONTINUE
      ID = -1
      PRINT 13
13    FORMAT(8H DCMODM,,5X,31HFAULT.   DISCRIMINANT SET TO -1)
      RETURN
      END
```

```
      SUBROUTINE DETMOD(A,IA,NN,IDET,MM)

C  PURPOSE,  COMPUTES DETERMINANT OF MATRIX A
C  ARGUMENTS,  A IS NN BY NN INTEGER MATRIX,  IA IS FIRST DIMENSION OF A IN
C              CALLING PROGRAMME.  IDET SET TO DETERMINANT, COMPUTED MOD MM.
C  RESTRICTIONS,  R*(MM)
C  CALLS  IJMODK,IZR,RECIPR
C  B-REGS,  0 - 11
C        IF DETERMINANT CANNOT BE COMPUTED, IDET = 0

      INTEGER A(IA,1)
      M = MM
      KS = 1
      DO 8 I = 1,NN
      DO 9 J = 1,NN
      KK = A(I,J)
      IF(KK) ,9,11
      A(I,J) = M-IZR(M,-KK)
      GO TO 9
11    A(I,J) = IZR(M,KK)
9     CONTINUE
8     CONTINUE
      DO 1 N = 1,NN-1
      IF(A(N,N)) 4,,4
      DO 5 I = N+1,NN
      IF(A(I,N)) 6,,6
5     CONTINUE
      IDET = 0
      PRINT 12
12    FORMAT(8H DETMOD,,5X,37HDETERMINANT NOT COMPUTED, SET TO ZERO)
      RETURN
6     DO 7 K = N,NN
      L = A(I,K)
      A(I,K) = A(N,K)
7     A(N,K) = L
      KS = -KS
4     CALL RECIPR(L,A(N,N),M,0)
      DO 2 I = N+1,NN
      LL = IJMODK(M,0,L,M-A(I,N))
      DO 3 K = N+1,NN
      J = A(I,K) + IJMODK(M,0,LL,A(N,K))
3     A(I,K) = IZR(M,J)
2     CONTINUE
1     CONTINUE
      L = IZR(M,KS+M)
      DO 10 N = 1,NN
10    L = IJMODK(M,0,L,A(N,N))
      IDET = L
      RETURN
      END
```

```
      SUBROUTINE DIFF(X,M,Y,N,A,NA,NUM,MM)

C  PURPOSE,  FORMS DIFFERENTIAL -DX/Y.
C  ARGUMENTS,  X INTEGER ARRAY Z**M [ ...,  Y INTEGER ARRAY Z**N [ ...
C              SETS INTEGER ARRAY A = -DX/Y = Z**NA [ A(1),...,A(NUM)]
C              MM IS MODULUS.
C  RESTRICTIONS,  (2,MM) = 1,  (Y(N),MM) = 1.  R+(MM)
C  CALLS  IZI,IZMR,IZR
C  B-REGS,  0 - 13
C        DX = 1/2.Z.DX/DZ

      INTEGER X(1),Y(1),A(1),W
      IX = IZI(2,MM)
      IXY = IZI(Y(N),MM)
      NA = M - N
      DO 1 I = M,NUM+M-1
      X(I) = IZMR  (MM,0,X(I),I+MM)
 1    X(I) = IZMR  (MM,0,X(I),MM-IY)
      DO 2 I = 1,NUM
      W = 0
      FOR J = 2,I
      W = W + IZMR  (MM,0,Y(N+J-1),A(I-J+1))
      REPEAT
      W = IZR(MM,W)
 2    A(I) = IZMR  (MM,0,IYY,X(M+I-1)-W+MM)
      RETURN
      END
```

```
      SUBROUTINE DISCUB(A1,A2,B1,B2,C1,C2,D,MM)

C  PURPOSE,  COMPUTES DISCRIMINANT OF CUBIC
C  ARGUMENTS,  CUBIC IS X**3 + A1/A2 X**2 + B1/B2 X + C1/C2,  WITH A2,B2,C2
C              NEGATIVE IF NECESSARY.  D IS SET TO DISCRIMINANT.  MM IS MODULUS
C  RESTRICTIONS,  M MUST BE RELATIVELY PRIME TO A2,B2,C2.  R*(M)
C  CALLS  IJMODK,IZI,IZR
C  B-REGS,  0 - 9
C         IF A = A1/A2, B = B1/B2,  C = C1/C2,  THEN

C                2 2     3       2                3
C         D = A B  - 4B  - 27C  + 18ABC - 4A  C

      INTEGER A1,A2,B1,B2,C1,C2
      INTEGER A,B,C,AA,AAA,BB,BBB,CC,D
      M = MM
      A = IJMODK(M,0,A1,IZI(A2*M,M))
      B = IJMODK(M,0,B1,IZI(B2*M,M))
      C = IJMODK(M,0,C1,IZI(C2*M,M))
      AA  = IJMODK(M,0,A,A)
      AAA = IJMODK(M,0,A,AA)
      BB  = IJMODK(M,0,B,B)
      BBB = IJMODK(M,0,B,BB)
      CC  = IJMODK(M,0,C,C)
      I = IJMODK(M,0,AA,BB)-IJMODK(M,0,4,BBB)-IJMODK(M,0,27,CC)
      AA = IJMODK(M,0,18,A)
      AA = IJMODK(M,0,AA,B)
      AA = IJMODK(M,0,AA,C)
      CC = IJMODK(M,0,4,AAA)
      CC = IJMODK(M,0,CC,C)
      I = I + AA + CC + M + M
      D = IZR(M,I)
      RETURN
      END
```

```
      SUBROUTINE EULD(I,J,N,M,MM)

C  PURPOSE,  CALLS EULDIV, EULDIE, QEULDI OR QEULDE, DEPENDING ON ARGUMENTS.
C  ARGUMENTS,  I,J INTEGER ARRAYS.
C              CALL EULD(I,J,N) WILL GIVE J = I/E(X) TO N TERMS, EXACTLY
C              CALL EULD(I,J,N,M) WILL CALL EULDIV WITH THESE ARGUMENTS IF M>0
C              , EULDIE IF M = 0.
C              CALL EULD(I,J,N,M,MM) WILL CALL QEULDI WITH THESE ARGUMENTS IF M
C             >0, QEULDE IF M = 0.
C       I(0),J(0) OVERWRITTEN.
C  RESTRICTIONS,  REUL(M,N), REULQ(M,N,MM),  R0(M)
C  CALLS EULDIV,EULDIE,QEULDI,QEULDE
C  B-REGS, 1,80 - 84

C       SUBROUTINE EULM(I,J,N,M,MM).  MUTATIS MUTANDIS, CALLING
C                                     EULMUL,EULMUE,QEULMU,QEULME

C              WRITTEN IN ASP
```

```
      SUBROUTINE EULDIV(I,J,N,M)

C  PURPOSE.  DIVIDES I = 1 + I(1) X + ... BY F(X) GIVING J = 1 + J(1) X + ...
C  ARGUMENTS.  M IS MODULUS.  N = NUMBER OF TERMS.  I(0),J(0) OVERWRITTEN.
C  RESTRICTIONS.  R*(M),REUL(M,N).
C        FORTRAN VERSION OF AN ASP ROUTINE.

      DIMENSION I(1),J(1)
      FOR L = 1,N
      JJ = I(L)
      K1 = L-1
      K3 = 1
      K2 = L-2
      K4 = -1
4     IF(K2) 2,2,3
3     K4 = K4-12
      JJ = JJ-J(K1)*J(K2)
      K3 = K3-12
      K2 = K2+K4
      K1 = K1+K3
      GO TO 4
2     IF(K2) 1,22,22
22    JJ = JJ-J(K1)*1
      GO TO 8
1     IF(K1) 8,6,7
6     JJ = JJ*1
      GO TO 8
7     JJ = JJ*J(K1)
8     K1 = L-5
      K3 = -5
      K2 = L-7
      K4 = -7
14    IF(K2) 12,12,13
13    K4 = K4-12
      JJ = JJ-J(K1)*J(K2)
      K3 = K3-12
      K2 = K2+K4
      K1 = K1+K3
      GO TO 14
12    IF(K2) 11,32,32
32    JJ = JJ*J(K1)*1
      GO TO 18
11    IF(K1) 18,16,17
16    JJ = JJ*1
      GO TO 18
17    JJ = JJ*J(K1)
18    J(L) = JJ-JJ/M*M
      IF(J(L)) 19,20,20
19    J(L) = J(L)+M
20    REPEAT
```

```
RETURN
END
```

```
      SUBROUTINE EULMUL(I,J,N,M)

C  PURPOSE.  MULTIPLIES I = 1 + I(1) X + ... BY F(X)  GIVING
C            J = 1 + J(1) X + ...
C  ARGUMENTS.  M IS MODULUS.  N = NUMBER OF TERMS.  I(0),J(0) OVERWRITTEN.
C  RESTRICTIONS.  R+(M),REUL(M,N).
C        TIME ~N**(3/2)
C        FORTRAN VERSION OF AN ASP ROUTINE.

      DIMENSION I(1),J(1)
      M6 = 6
      M5 = 5
      DO 20 L = 1,N
      JJ = I(L)
      K = 1
      K1 = L + 1
 4    K2 = K1 + K
      IF(K2) 2,2,
      JJ = JJ + I(K1) * I(K2)
      K1 = K1 + K*M6 - M5
      K = K + 2
      GO TO 4
 2    IF(K2) 1,,
      JJ = JJ + I(K1) * 1
      GO TO 8
 1    IF(K1) 8,,7
      JJ = JJ + 1
      GO TO 8
 7    JJ = JJ + I(K1)
 8    K = 2
      K1 = L + M5
 14   K2 = K1 + K
      IF(K2) 12,12,
      JJ = JJ + I(K1) * I(K2)
      K1 = K1 + K*M6 - M5
      K = K + 2
      GO TO 14
 12   IF(K2) 11,,
      JJ = JJ + I(K1) * 1
      GO TO 18
 11   IF(K1) 18,,17
      JJ = JJ + 1
      GO TO 18
 17   JJ = JJ + I(K1)
 18   J(L) = JJ - JJ/M*M
      IF(J(L)) ,20,20
      J(L) = J(L) + M
 20   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FACTOR(SS,NN,JF,L,IPRINT)
C  PURPOSE,  FACTORISATION ROUTINE.
C  ARGUMENTS.  NUMBER TO BE FACTORISED IS SS(1)*10**4**(NN-1) + ... + SS(NN).
C              IF NN>3, HAVE SS(I) < 10**4.  IF NN = 0, NUMBER TO BE
C              FACTORISED IS SS = SS(1).  NUMBER MADE POSITIVE BEFORE BEING
C              FACTORISED AS  PRODUCT_{N=1}^{L/2} JF(2N-1)**JF(2N).  ATTEMPTS TO FACTORISE
C              USING PRIMES < 10**4 WHICH MUST BE AT BLOCK 1 ON TAPE 2.
C              IF IPRINT NEQ 0, FACTORS ARE PRINTED - * OR ** INDICATES A NUMBE
C              WITH NO PRIME FACTOR < 10**4
C  RESTRICTIONS.  NUMBER TO BE FACTORISED < 10**1000.  NN = 0 OR 3≤NN≤250.
C  CALLS  IJMODK,IZR,PRIFAC
C  B-REGS.  0 - 10
C        IF NUMBER TO BE FACTORISED < 10** 100, HAVE DIM. JF(108) IN MAIN PROG.
C        IF NUMBER TO BE FACTORISED < 10** 200, HAVE DIM. JF(186) IN MAIN PROG.
C        IF NUMBER TO BE FACTORISED < 10** 300, HAVE DIM. JF(258) IN MAIN PROG.
C        IF NUMBER TO BE FACTORISED < 10** 400, HAVE DIM. JF(326) IN MAIN PROG.
C        IF NUMBER TO BE FACTORISED < 10** 500, HAVE DIM. JF(392) IN MAIN PROG.
C        IF NUMBER TO BE FACTORISED < 10** 600, HAVE DIM. JF(456) IN MAIN PROG.
C        IF NUMBER TO BE FACTORISED < 10** 700, HAVE DIM. JF(520) IN MAIN PROG.
C        IF NUMBER TO BE FACTORISED < 10** 800, HAVE DIM. JF(582) IN MAIN PROG.
C        IF NUMBER TO BE FACTORISED < 10** 900, HAVE DIM. JF(642) IN MAIN PROG.
C        IF NUMBER TO BE FACTORISED < 10**1000, HAVE DIM. JF(702) IN MAIN PROG.

      INTEGER SS(1),JF(1)
      INTEGER J(1229),J1230,S(250),MM(250)
      IF(III) 1,,1
      MM(1),III = 1
      J1230 = 10007          <-- M = 10000
      REWIND 2
      READ TAPE 2,J
      IF((J(1).EQ.2).AND.(J(1229).EQ.9973)) 1,
      PRINT 23,J(1),J(1229)
23    FORMAT(8H FACTOR.,5X,34HINCORRECT LIST OF PRIMES - J(1) = ,I12,
     1 12H, J(1229) = ,I12)
      RETURN
1     LINSTAR,JPREV = 0
      JJJ = 1
      IF(NN) 28,,8
      N = IABSF(SS(1))
      IF(N - 100140048) ,,9
34    I = 1
      GO TO 18
9     S(3) = N/100000000
      S(2) = N/10000 - 10000*S(3)
      S(1) = IZR(M,N)
      N = 3
      GO TO 10
```

IF(N-2) ,,34
JF(L+1) = N
L = L+2
JF(L) = 1
GO TO 25

```
8     N = NN
      IF(N-250) 27,27,
28    PRINT 29
29    FORMAT(8H FACTOR,,5X,36HNUMBER HAS TOO MANY (OR TOO FEW) DIGITS,
     1 INCREASE ARRAYS S AND MM TO CURE FIRST CASE)
      RETURN
27    DO 2 I = 1,N
2     S(I) = SS(N+I+1)
      S(N) = IABSF(S(N))
10    DO 3 I = 1,1229
      JJ = J(I)
      DO 4 K = 2,N
4     MM(K) = IJMODK(JJ,0,M,MM(K-1))
15    LL = 0
      DO 5 K = 1,N
5     LL = LL + IJMODK(JJ,0,S(K),MM(K))
      LL = IZR(JJ,LL)
      IF(LL) 3,,3
      IF(JPREV) 11,,11
      JPREV = JJ
      GO TO 14
11    IF(JJ-JPREV) ,12,
      JF(L+1) = JPREV
      L = L + 2
      JF(L) = JJJ
      JPREV = JJ
      JJJ = 0
12    JJJ = JJJ + 1
14    DO 6 K = N,3,-1
      KK = S(K)/JJ
      S(K-1) = S(K-1) + (S(K) - KK*JJ)*M
6     S(K) = KK
      KK = S(2)/JJ
      S(1) = (S(1) + S(2)*M)/JJ - KK*M
      S(2) = KK
      NNN = N
      DO 7 K = NNN,3,-1
      IF(S(K)) ,,15
      N = N-1
7     CONTINUE
      N = S(2)*M + S(1)
      GO TO 18
3     CONTINUE
      NSTAR = 1
      GO TO 20
18    II = N/J(I)
      IF(N-J(I)*II) ,16,
      I = I + 1
      GO TO 17
16    IF(JPREV) 13,,13
      JPREV = J(I)
```

```
      GO TO 19
   13 IF(J(I) - JPREV) ,21,
      JF(L + 1) = JPREV
      L = L + 2
      JF(L) = JJJ
      JPREV = J(I)
      JJJ = 0
   21 JJJ = JJJ + 1
   19 N = II
   17 IF(J(I)*J(I) - N) 18,18,
      IF(JPREV) 22,,22
      JPREV = N
      GO TO 20
   22 IF(N - JPREV) ,24,
      JF(L + 1) = JPREV
      L = L + 2
      JF(L) = JJJ
      JPREV = N
      JJJ = 0
   24 JJJ = JJJ + 1
   20 IF(JPREV) ,25,
      JF(L + 1) = JPREV
      L = L + 2
      JF(L) = JJJ
   25 IF(IPRINT) ,26,
      NCOL = 5
      CALL PRIFAC(JF,L,NCOL,NSTAR)
      IF(NCOL - 150) 30,,
      PRINT 31,(JF(I),I = 1,L)
   31 FORMAT(5(1X,I11,I6))
      IF(NSTAR) ,32,
      PRINT 33
   33 FORMAT(3H **)
   32 RETURN
   30 CALL OUTREC
   26 RETURN
      END
```

```
      SUBROUTINE FINDRT(LQ,LPOWER,LNUM,LDEN,LROOT,LANS)

C  PURPOSE,  CALCULATES LANS FROM LANS**LROOT = LNUM/LDEN (MOD LQ**LPOWER)
C            IF LANS NOT FOUND , LANS = 1
C  RESTRICTIONS,  R*(LQ**LPOWER),  LROOT ≥ 2
C  CALLS  IZI,IZMR,IZR
C  B-REGS,  0 - 9

      MQ = LQ**LPOWER
      LX = IZMR(MQ,0,IZI(LDEN,MQ),LNUM)
      LXX = IZR(LQ,LX)
      DO 1 LI = 1,LQ-1
      LX = LI
      DO 2 LL = 1,LROOT-1
    2 LX = IZMR(LQ,0,LX,LY)
      IF(LY - LXX),,4
    1 CONTINUE
      PRINT 5
    5 FORMAT(8H FINDRT,,5X,26HNO ROOT FOUND  -- LANS = 1)
      LANS = 1
      RETURN
    4 LANS = LI
      LRT = LROOT*LXX
      LF = IZMR(LQ,0,IZI(LRT,LQ),LANS)
      FOR L = 1,LPOWER-1
      LY = LANS
      DO 6 LL = 1,LROOT-1
    6 LX = IZMR(MQ,0,LY,LANS)
      LZ = (LX-LY*MQ)/(LQ**L)
      LAMB = IZMR(LQ,0,LZ,LF)
      LANS = LANS + LAMB*(LQ**L)
      REPEAT
      RETURN
      END
```

```
      FUNCTION IVAL(I,LP,MAX)

C  PURPOSE,  IVAL = POWER OF LP LE MAX DIVIDING I.  IF THIS IS MAX,  IVAL = -1.
C  RESTRICTIONS,  RMOD(LP**MAX,I)
C  CALLS  IZR
C  B-REGS.  0 - 5

      LL = 1
      DO 1 L = 1,MAX
      LL = LP*LL
      LLL = IZR(LL,I)
      IF(LLL) 2,,2
1     CONTINUE
      IVAL = -1
      RETURN
2     IVAL = L-1
      RETURN
      END
```

```
      FUNCTION IZI(IX,M)

C  PURPOSE,  RECIPROCAL OF IX MOD M
C  ARGUMENTS,  M IS MODULUS
C  RESTRICTIONS,  R*(M)
C  CALLS IZI,IZQ,IZR
C  B-REGS,  0 - 5
C        IF NO RECIPROCAL, IZI = 0.
C        IY = IZI(IX,M) IS EQUIVALENT TO CALL RECIPR(IY,IX,M,0)

      DIMENSION I(40),K(40)
      K1=IX
      K2=M
      K1=K1-K1/M*M
      IF(K1) ,4,1
      K1=K1+M
1     L=1
2     I(L) = IZQ(0,0,0,K1,K2)
      J=K2-K1*I(L)
      IF(J) ,3,
      L = L + 1
      K2=K1
      K1=J
      GO TO 2
3     IF (K1-1) ,5,
4     IZI = 0
      RETURN
5     IF(L-2) 6,,6
      IZI = M - I(1)
      RETURN
6     IF(L-1) ,,7
      IZI = 1
      RETURN
7     K(L)=1
      K(L-1) =I(L-1)
      DO 8  LL=L-2,1,-1
8     K(LL)=K(LL+2)-K(LL+1)*I(LL)
      LL = IZR(2,L)
      IF(LL) 9,,9
      K(1) = M - K(1)
9     IZI = K(1)
      RETURN
      END
```

```
      FUNCTION IZIU(IX,M,IPRINT)

C  PURPOSE,  COMPUTES RECIPROCAL OF IX MOD M
C  ARGUMENTS,  M IS MODULUS.  PRINTS RECIPROCAL IF IPRINT NEQ 0.
C  RESTRICTIONS,  R*(M),RPRINT
C  CALLS  IZ0,IZR
C  B-REGS, 0 = 6
C         IF NO RECIPROCAL ,IZIU = HCF OF IX AND M.
C         IY = IZIU(IX,M,IPRINT) IS EQUIVALENT TO CALL RECIPR(IY,IX,M,IPRINT)

      DIMENSION I(40),K(40)
      K1=IX
       K2=M
      K1=K1-K1/M*M
      IF(K1) ,4,1
      K1=K1+M
 1    L=1
 2    I(L) = IZ0(0,0,0,K1,K2)
       U=K2-K1*I(L)
      IF(U) ,3,
      L = L + 1
       K2=K1
       K1=U
      GO TO 2
 3    IF (K1-1) ,5,
 4    IZIU = K1
      PRINT 10,IX,M,K1
 10   FORMAT(8H HCF OF ,I12,5H AND ,I12,4H IS ,I12,10X,21HIZIU SET EQUAL
     1 TO HCF)
      RETURN
 5    IF(L-2) 6,,6
      IZIUU = M - I(1)
      IZIU = IZIUU
      GO TO 13
 6    IF(L-1) ,,7
      IZIUU = 1
      IZIU = IZIUU
      GO TO 13
 7     K(L)=1
      K(L-1) =I(L-1)
      DO 8  LL=L-2,1,-1
 8    K(LL)=K(LL+2)+K(LL+1)*I(LL)
      LU = IZR(2,L)
      IF(LL) 9,,9
      K(1) = M - K(1)
 9    IZIUU = K(1)
      IZIU = IZIUU
 13   IF(IPRINT) ,11,
      PRINT 12,M,IX,IZIUU
 12   FORMAT(8H MODULO ,I12,15H RECIPROCAL OF ,I12,4H IS ,I12,8H (=IZIU)
```

```
     1)
11     RETURN
       END
```

```
      FUNCTION IZMQ(0,0,0,M,0,I,J)
C
C  PURPOSE,  IZMQ SET TO QUOTIENT (I*J)/M.
C  ARGUMENTS.  FIRST,SECOND,THIRD AND FIFTH ARGUMENTS ARE ''DUMMY'',
C              CONVENIENTLY 0.
C  RESTRICTIONS.  R*(M),RMOD(M,I,J)
C  B-REGS.  1 - 3.
C        OVERWRITES CALLING SEQUENCE.
C        WRITTEN IN ASP
```

```
      FUNCTION IZMR(K,0,I,J)
C PURPOSE,  IZMR SET EQUAL TO I*J (MOD K)
C ARGUMENTS,  0 ≤ IZMR < K, SECOND ARGUMENT ;;DUMMY;; , CONVENIENTLY ZERO
C RESTRICTIONS,  RMOD(K,I,J),RIF
C B-REGS,  1 - 3
C       OVERWRITES CALLING SEQUENCE
C       IDENTICAL TO IJMODK
C       WRITTEN IN ASP
```

```
      FUNCTION IZMRU(M,I,J)

C  PURPOSE:  GIVES I*J MOD M
C  RESTRICTIONS,  R*(M)
C  CALLS IZMR,IZR
C  B-REGS:  0 - 4

      IF(I) 1,,2
 3    IZMRU = 0
      RETURN
 1    K = M-IZR(M,-I)
      IF(K-M) 4,3,4
 2    K = IZR(M,I)
 4    IF(J) ,3,5
      L = M-IZR(M,-J)
      IF(L-M) 6,3,6
 5    L = IZR(M,J)
 6    IZMRU = IZMR(M,0,K,L)
      RETURN
      END
```

```
      FUNCTION IZP(I,N,M)

C  PURPOSE,  SETS IZP = I**N (MOD M)
C  ARGUMENTS, IF I = 0, IZP = 0
C  RESTRICTIONS,  R+(M), -2**35 ≤ I,N ≤ 2**35
C  CALLS  IZI
C  B-REGS,  1,41 - 50,124
C        WRITTEN IN ASP.
```

```
      FUNCTION IZPR(N)
C  PURPOSE ,  CALCULATES LEAST PRIMITIVE ROOT OF A PRIME.
C  ARGUMENTS.  N MUST BE A PRIME GE 2
C  CALLS  FACTOR,IZP,IZQ
C  B-REGS.  0 - 6
      DIMENSION JF(20)
      IF(N-3) ,,5
      IF(N-1) 6,6,
      IZPR = N-1
      RETURN
 5    NN = N
      M = NN-1
      CALL FACTOR(M,0,JF,L,0)
      L = L/2
      DO 1 I = 1,L
 1    JF(I) = IZQ(0,0,0,JF(2*I-1),M)
      DO 2 I = 2,M
      DO 3 J = 1,L
      K = IZP(I,JF(J),NN)
      IF(K-1) ,2,
 3    CONTINUE
      IZPR = I
      GO TO 4
 2    CONTINUE
 6    IZPR = 0
 4    RETURN
      END
```

```
C
      FUNCTION IZQ(0,0,0,M,I)
C PURPOSE  IZQ SET EQUAL TO I/M
C ARGUMENTS.  FIRST,SECOND AND THIRD ARGUMENTS DUMMY,CONVENIENTLY ZERO.
C RESTRICTIONS.  R*(M),RO(I),RIF
C B-REGS.  1 - 3,124
C      OVERWRITES CALLING SEQUENCE
C      WRITTEN IN ASP
```

```
      FUNCTION IZQU(M,I)
C     PURPOSE,  GIVES I/M
C     RESTRICTIONS,  R*(M)
C     B-REGS,  0 - 3
      IZQU = I/M
      RETURN
      END
```

```
C
      FUNCTION IZR(M,I)
C  PURPOSE,  IZR SET EQUAL TO I MOD M.
C  RESTRICTIONS,  R*(M),R0(I),RIF
C  B-REGS,  1 - 3,124
C        OVERWRITES CALLING SEQUENCE
C        WRITTEN IN ASP
```

```
      FUNCTION IZRU(M,I)
C  PURPOSE,  GIVES I MOD M
C  RESTRICTIONS,  R+(M)
C  CALLS  IZR
C  B-REGS,  0 - 3
      IF(I) 1,,2
 3    IZRU = 0
      RETURN
 1    J = M+IZR(M,-I)
      IF(J-M) ,3,
      IZRU = J
      RETURN
 2    IZRU = IZR(M,I)
      RETURN
      END
```

```
      SUBROUTINE KLEINJ(I,N,M,J123)
C  PURPOSE,  PLANTS J,√(J-1728) OR ∛J IN I, ACCORDING AS J123 = 1,2 OR 3.
C  ARGUMENTS.  I INTEGER ARRAY, N NUMBER OF TERMS COMPUTED, M IS MODULUS.
C              EG, IF J123 = 1, I(1) = 1,  I(2) = 744 (MOD M),  I(3) = 196884
C              (MOD M),...     I(0) OVERWRITTEN.
C  RESTRICTIONS, FOR CORRECT END AT TOP OF SERIES, MAKE N ZERO MOD J123.
C               M MUST NOT BE ZERO MOD 691.  REUL(M,N),R+(M),K(N)*M < F,
C               WHERE K(N) = D(H) AND H IS LARGEST HIGHLY COMPOSITE NUMBER ≤ N
C  CALLS  EULDIV,IZI,IZMR,IZR
C  B-REGS,  0 = 10
C        LIST OF HIGHLY COMPOSITE NUMBERS N
C           N  D(N)          N  D(N)          N  D(N)
C
C           2    2         240   20        7560   64
C           4    3         360   24       10080   72
C           6    4         720   30       15120   80
C          12    6         840   32       20160   84
C          24    8        1260   36       25200   90
C          30    9        1680   40       27720   96
C          48   10        2520   48       45360  100
C          60   12        5040   60       50400  108
C         120   16
C         180   18
      DIMENSION I(1)
      DO 1 L=1,N
1     I(L)=0
      IF(J123-2) ,6,7
      LC = IZMR(M,0,65520,IZI(691,M))
      LS1=10
      LEULER=24
      GO TO 8
6     LC=M*504+504/M*M
      LS1=4
      LEULER=12
      GO TO 8
7     LC=240
      LS1=2
      LEULER = 8
8     DO 99 L=1,N
      DO 100 LL=L,N,L
      LX=L
      DO 101 LS=1,LS1
101   LX = IZMR(M,0,L,LX)
100   I(LL) = I(LL) * LX
99    CONTINUE
      DO 2 L=1,N
2     I(L)=IZMR  (M,0,LC,I(L))
      DO 3 L=1,LEULER
3     CALL EULDIV(I,I,N,M)
```

```
      DO 4 L=N,2,-1
4     I(L)=I(L-1)
      IF (J123-2)    ,46,46
      J(2) = IZR(M,744)
46    J(1)=1
      IF(J123 - 2) 10,,12
      DO 13 L=N/2,1,-1
      J(2*L-1)=I(L)
13    J(2*L)=0
      GO TO 10
12    DO 14 L= N/3,1,-1
      J(3*L-2)=I(L)
14    I(3*L-1),I(3*L)=0
10    RETURN
      END
```

```
      SUBROUTINE MODCOF(Q,MT,XX,YY,D1,D2,DD1,DD2,DDD1,DDD2,K1,K2,KLN,MM)
C
C  PURPOSE,  COMPUTES ARRAYS XX = X, YY = Y FROM RELATIONS GIVEN BELOW.
C  ARGUMENTS,  MM IS MODULUS,
C        Q INDEX (GAMMA Q)    Q<51
C        MT = GIVES XX,YY, UP TO AND INC. TERM IN Z**MT
C        D1/D2 ETC, ARE COEFFS (DENOM NEG IF NEC)
C        K**Q = K1/K2        (NUMERATOR NEG IF NEC)
C        KLN = 1 FOR J
C              2 FOR SQRT(J+1728)
C              3 FOR CUBE  ROOT J.
C        MM = MODULUS
C        Y**2 = X**3 + D1(1)/D2(1)X**2 + D1(2)/D2(2)X + D1(3)/D2(3)
C        J123 = Y*(DD1(1)/DD2(1)X**N + ...) + DDD1(1)/DDD2(1)X**(N+2)+...
C        WHERE
C        Q EVEN , N = (Q-4)/2    DDD1(1)/DDD2(1) = 1
C        Q ODD  , N = (Q-3)/2    DD1(1)/DD2(1) = 1,  DDD1(1)/DDD2(1) = 0
C  RESTRICTIONS,  R*(MM), Q < 51
C  CALLS  COEFF,IJMODK,KLEINJ,RECIPR
C  B-REGS,  0 = 20
C        SEE SECTION 2,4.

      INTEGER D1(1),D2(1),DD1(1),DD2(1),DDD1(1),DDD2(1),XX(1),YY(1)
      INTEGER C(4),CC(26),CCC(26)
      INTEGER D3(50),X01(2000),D4(50),X02(2000),D5(50),X( 8,2000),D6(50)
     1,Y1(2000),D7(50),Y2(2000),D8(50),YX( 8,2000),D9(50),AJ(2001),
     2 FC(250),W1(3),W(2000),WW1(50),WW(2000)
      INTEGER A,AK,AAK,B,Q,Q6,RECIP,S,S1,S2,SS1,SS2
      NTT = MT
      NN = NTT*2
      Q6 = Q-6
      NN = (Q-3)/2
      NNN = NTT/Q+2
      CALL RECIPR(INV2,2,MM,0)
C
C        ZEROISE ARRAYS.
      DO 1000 I = 1,50
 1000 D5(I),D6(I),D7(I),D8(I) = 0
      DO 1001 I = 1,NTT
 1001 Y1(I),Y2(I) = 0
      DO 1002 I = 1,NN*2
      DO 1003 II = 1,NTT*1
 1003 X(I,II),YX(I,II) = 0
 1002 CONTINUE
C
C        EVALUATE C,CC,CCC
      DO 1 I = 1,3
      CALL RECIPR(II,D2(4-I)*MM,MM,0)
    1 C(I) = IJMODK(MM,0,D1(4-I),II)
      C(4) = 1
```

```
      DO 4 I = 1,NN+1
      CALL RECIPR(II,DD2(NN+2-I)+MM,MM,0)
 4    CC(I) = IJMODK(MM,0,DD1(NN+2-I),II)
      DO 5 I = 1,NN+3
      CALL RECIPR(II,DDD2(NN+4-I)+MM,MM,0)
 5    CCC(I) = IJMODK(MM,0,DDD1(NN+4-I),II)
C
      IF(Q - Q/2*2) 2,,2
      SS1 = 0
      CALL RECIPR(SS2,MM-(NN+2),MM,0)
      CCC(NN+3) = 1      * DATA SHOULD GIVE THIS
      GO TO 3
 2    CALL RECIPR(SS1,MM-Q,MM,0)
      SS2 = IJMODK(MM,0,SS1,2)
      CC(NN+1) = 1       * DATA SHOULD GIVE THIS
      CCC(NN+3) = 0      * DATA SHOULD GIVE THIS
C        SET UP J(KLN)
 3    CALL KLEINJ(FC,(NNN-1)/6*6+6,MM,KLN)
      CALL RECIPR(II,K1+MM,MM,0)
      AK = IJMODK(MM,0,K2,II)
      CALL RECIPR(AAK,AK,MM,0)
      DO 6 I = 1,NNN
      AAK = IJMODK(MM,0,AAK,AK)
 6    AA(Q*(I-2)) = IJMODK(MM,0,FC(I),AAK)
      X01(0) = C(1)
      X02(0) = CCC(1)
      DO 7 I = 1,NN+2
 7    X(I,-2*I) = 1
      DO 8 I = 1,NN
 8    YX(I,-2*I+3) = 1
      Y1(-3),Y2(-6) = 1
      DO 9 N = -5,NTT - 3
      CALL COEFF(Y1,-3,Y1,-3,N,Y2(N),MM)
      DO 210 I = -2,N+4
210   W(I) = X(1,I)
      CALL COEFF(W,-2,W,-2,N+2,X(2,N+2),MM)
      DO 10 I = 2,NN+1
      DO 211 L = -2*I,N-2*I+6
211   WW(L) = X(I,L)
 10   CALL COEFF(W,-2,WW,-2*I,N-2*I+4,X(I+1,N-2*I+4),MM)
      CALL COEFF(W,-2,Y1,-3,N+1,YX(1,N+1),MM)
      DO 11 I = 2,NN
      DO 212 L = -2*I,N-2*I+6
212   WW(L) = X(I,L)
 11   CALL COEFF(WW,-2*I,Y1,-3,N-2*I+3,YX(I,N-2*I+3),MM)
      S = 0
      DO 12 I = 1,3
 12   S = S + IJMODK(MM,0,C(I+1),X(I,N))
      S1 = X01(N)+S+Y2(N)+MM
C
C
```

```
      N1 = N - Q6
      S = IJMODK(MM,0,CC(1),Y1(N1))
      DO 13 I = 1,NN
   13 S = S + IJMODK(MM,0,CC(I+1),YX(I,N1))
      DO 14 I = 1,NN+2
   14 S = S + IJMODK(MM,0,CCC(I+1),X(I,N1))
      S2 = S + X02(N1) + AJ(N1)+MM
C
      S1 = IJMODK(MM,0,1,S1)
      S2 = IJMODK(MM,0,1,S2)
      A = IJMODK(MM,0,SS1,S1) + IJMODK(MM,0,SS2,S2)
      B = IJMODK(MM,0,3*A+S1,INV2)
      X(1,N+4) = A
      K = N+4
      DO 15 I = 2,NN+2
      K = K - 2
   15 X(I,K) = X(I,K) + I*A
      Y1(N+3) = B
      Y2(N) = Y2(N)+2*B
      K = N+3
      DO 16 I = 1,NN
      K = K - 2
   16 YX(I,K) = YX(I,K) + I*A + B
    9 CONTINUE
      DO 17 I = -2,NTT
   17 XX(I) = IJMODK(MM,0,1,X(1,I))
      DO 18 I = -3,NTT
   18 YX(I) = Y1(I)
      RETURN
      END
```

```
      SUBROUTINE QEULMU(I,J,N,M,MM)

C  PURPOSE.   MULTIPLIES I = 1 + I(1) X + ...  BY F(X**MM)  GIVING J = 1 + J(1)
C                + ...
C  ARGUMENTS.  M IS MODULUS.  N = NO. OF TERMS
C  RESTRICTIONS.  R=(M),REULQ(M,N,NN) , J MUST NOT BE ARRAY I
C        FORTRAN VERSION OF AN ASP ROUTINE.

      DIMENSION I(1),J(1)
      M6 = 6*MM
      M5 = 5*MM
      DO 20 L = 1,N
      JJ = I(L)
      K = 1
      K1 = L + MM
 4    K2 = K1 + K*MM
      IF(K2) 2,2,
      JJ = JJ - I(K1) - I(K2)
      K1 = K1 + K*M6 - M5
      K = K + 2
      GO TO 4
 2    IF(K2) 1,,
      JJ = JJ - I(K1) - 1
      GO TO 8
 1    IF(K1) 8,,7
      JJ = JJ - 1
      GO TO 8
 7    JJ = JJ - I(K1)
 8    K = 2
      K1 = L + M5
 14   K2 = K1 + K*MM
      IF(K2) 12,12,
      JJ = JJ + I(K1) + I(K2)
      K1 = K1 + K*M6 - M5
      K = K + 2
      GO TO 14
 12   IF(K2) 11,,
      JJ = JJ + I(K1) + 1
      GO TO 18
 11   IF(K1) 18,,17
      JJ = JJ + 1
      GO TO 18
 17   JJ = JJ + I(K1)
 18   J(L) = JJ - JJ/M*M
      IF(J(L)) ,20,20
      J(L) = J(L) + M
 20   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE READML(FCOL,NCARD,LCOL,NPRINT,JF,L)

C  PURPOSE,  READS AN INTEGER OUTPUT BY A ML PRINT STATEMENT AND ATTEMPTS TO
C            FACTORISE IT.
C  ARGUMENTS.  FIRST DIGIT OF NUMBER IS ON CARD 1 COLUMN FCOL, LAST DIGIT OF
C              NUMBER IS ON CARD NCARD, COLUMN FCOL.  IF NPRINT NEQ 0 WILL PRIN
C              NUMBER.  FACTORS LEFT IN ARRAY JF(I),I = 1,L AS IN FACTOR.
C              FACTORS ARE PRINTED.
C  RESTRICTIONS NUMBER MUST BE POSITIVE AND HAVE GREATER THAN 8 AND LESS THAN
C               1000 DIGITS.
C  CALLS  FACTOR
C  B-REGS.  0 - 11

      INTEGER JF(1)
      INTEGER FCOL,SS(3),S(1300)
  100 FORMAT(80I1)
  101 FORMAT(1X,I4,4Z,29I4)
      M = 10
      NC = 80*NCARD
      READ 100,(S(I),I = 1,NC)
      NINC = NC+LCOL-FCOL-79
      NGRP5 = NINC/6
      NFRONT = NINC-6*NGRP5          * NFRONT = 1,2,3,4 OR 5.
      NN = 0
      DO 1 I = 1,NFRONT
      NN = NN + 1
    1 S(NN) = S(FCOL+I-1)
      MM = FCOL+NFRONT-6
      DO 2 I = 1,NGRP5
      MM = MM+6
      DO 3 J = 1,5
      NN = NN + 1
    3 S(NN) = S(MM+J)
    2 CONTINUE
C          NN DIGITS
      NGRP4 = NN/4
      NFRONT = NN-4*NGRP4            * NFRONT = 0,1,2 OR 3.
      J = 1
      IF(NFRONT) ,4,
      DO 5 I = 1,4-NFRONT
      J = J + 1
    5 S(J) = 0
      NGRP4 = NGRP4 + 1
    4 J = J - 4
      DO 6 I = 1,NGRP4
      J = J + 4
    6 S(I) = ((S(J)*M+S(J+1))*M+S(J+2))*M+S(J+3)
      IF(NPRINT) ,7,
      PRINT 101,(S(I),I = 1,NGRP4)
      CALL IOZ(1)
```

```
7       CALL FACTOR(S,NGRP4,JF,L,1)
        RETURN
        END
```

```
      SUBROUTINE SUMRES(I,IN,J,JN,IANS)
C
C  PURPOSE.  COMPUTES CHARACTER SUMS.
C  ARGUMENTS.  I,J,IANS  INTEGER ARRAYS.
C              CURVE IS
C                Y**2 = J(1) X**(JN-1) + ... + J(JN).
C              FOR L = 1,IN SETS IANS(L) = SUM OVER RESIDUE CLASS MOD P OF
C              LEGENDRE SYMBOL (Y**2 / P)  WHERE P = I(L), A PRIME.
C  RESTRICTIONS.  I(L) < 1000,  JN >= 2,  R+(I(L))
C  CALLS  IZMR,IZR
C  B-REGS.  0 - 10
C
      DIMENSION I(1),J(1),IANS(1),K(100),KK(1000)
      DO 1 L = 1,IN
      LP = I(L)
      KK(1) = 0
      DO 3 LL = 2,LP
    3 KK(LL) = -1
      DO 7 LLL = 1,(LP-1)/2
      LL = IZMR(LP,0,LLL,LLL)
    7 KK(LL+1) = 1
      DO 2 LL = 1,JN
      IF(J(LL)) 4,2,4
      K(LL) = LP - IZR(LP,-J(LL))
      GO TO 2
    4 K(LL) = J(LL)
    2 CONTINUE
      KKK = 0
      DO 5 LL = 0,LP-1
      KL = K(1)
      DO 6 LLL = 2,JN
    6 KL = IZMR(LP,0,LL,KL) + K(LLL)
      KL = IZR(LP,KL)
    5 KKK = KKK + KK(KL+1)
    1 IANS(L) = KKK
      RETURN
      END
```

```
C         SUBROUTINES MLO,MLT,MLD,SL,TO,TOSL,PREC,ADD,SUB,MULT,DIV,COMP,
C                     RCP,ROUND,IPART,FPART,POS,SQROOT,SQ,READ,POWER,
C                     MODULO,PRINT,NP,SP,TEXT,NL,PRECAS,SIZE,NAMES,.

C  PURPOSE.  MULTI-LENGTH ARITHMETIC PACKAGE.
C  CALLS  /ML
C  B-REGS.  1,56 - 63,65 - 69,81 - 85,91,92,97,124
C         THESE ROUTINES ARE WRITTEN IN ASP.
```

```
      SUBROUTINE DC(A,N,MM,MMM,X,P)

C  PURPOSE,  COMPUTES DISCRIMINANT USING CHINESE REMAINDER THEOREM.
C  ARGUMENTS,  A INTEGER ARRAY,  ML=VARIABLE X SET TO DISCRIMINANT OF
C              A(1) X**N + ... + A(N+1),  USING MMM MODULI MM(I),I = 1,MMM.
C              P IS PRECISION.
C  RESTRICTIONS,  N <= 26,  R*(MM(I)),I = 1,MMM
C  CALLS  DETMOD,IJMODK,IZR,MLCREM
C  B-REGS,  0 - 12

      INTEGER P,B(49,49),A(1),MM(1),DET(50),AA0(26),AA(26),AA1(26)
      N1 = N + 1
      N2 = 2*N - 1
      DO 5 I = 1,26
 5    AA0(I),AA(I),AA1(I) = 0
      DO 1 L = 1,MMM
      M = MM(L)
      DO 2 I = 1,N1
      IF(A(I)) 2,2,3
      AA(I) = M-IZR(M,-A(I))
      GO TO 2
 3    AA(I) = IZR(M,A(I))
 2    CONTINUE
      K = 1
      DO 4 I = 2,N2,2
      K = K+1
      DO 6 J = 1,N2
 6    B(I,J) = AA(J+K)
 4    CONTINUE
      AA(N1) = 0
      DO 7 I = 1,N+1
 7    AA(I) = IJMODK(M,0,N1-I,AA(I))
      K = 1
      DO 8 I = 1,N2,2
      K = K+1
      DO 9 J = 1,N2
 9    B(I,J) = AA(J+K)
 8    CONTINUE
      B(1,1) = N
      B(2,1) = 1
 1    CALL DETMOD(B,49,N2,DET(L),M)
      CALL MLCREM(DET,MM,MMM,X,P)
      RETURN
      END
```

```
      SUBROUTINE MLCF(X,Y,P)

C  PURPOSE.  OUTPUTS CONTINUED FRACTION.
C  ARGUMENTS.  X,Y  ML-VARIABLES.  P PRECISION.  OUTPUTS CONTINUED FRACTION OF
C              X UNTIL CORRESPONDING PARTIAL QUOTIENTS OF X AND Y DIFFER.
C              NORMALLY Y = X * 10**(-P+5),SAY
C  CALLS  ML
C  B-REGS.  0 - 4

      EXTERNAL IPART,FPART,RCP
      CALL MLD(PREC(P),NAMES(Z,I))
      CALL MLO(X, IPART, TO(I), NL(2), PRINT(10,0), X, FPART, TO(X))
      CALL MLO(Y, TO(Z), FPART, TO(Y))
      IF (MLT(Z, IPART, SUB(I))) 202, ,202
      CALL MLO(NL(1))
  200 IF (MLT(X)) ,201,
      CALL MLO(X, RCP, TO(X), IPART, TO(I), PRINT(10,0), X, FPART,TO(X))
      IF (MLT(Y)) ,203,
      IF (MLT(Y,RCP,TO(Z),FPART,TO(Y),Z,IPART,SUB(I))) 202,200,202
  201 CALL MLO( TEXT(12H           *))
      IF (MLT(Y)) ,204,
      CALL MLO(Y, RCP, TO(Z))
  202 CALL MLO( TEXT(1H,), Z, IPART, PRINT(1,0))
      GO TO 204
  203 CALL MLO( TEXT(2H,*))
  204 RETURN
      END
```

```
      SUBROUTINE MLCOS(X,Y,P)

C  PURPOSE.  COMPUTES ML COS FUNCTION
C  ARGUMENTS.  X,Y ML-VARIABLES.  P PRECISION.  SETS Y = COS(X)
C  CALLS  ML
C  B-REGS.  0 - 4
C         READS # FROM CURRENT INPUT STREAM ON FIRST EXECUTION.

      INTEGER P
      EXTERNAL READ,SQ,POS
      IF(III)1,,1
      III = 1
      CALL MLD(PREC(P),NAMES(EPS,PI,F,T))
      CALL MLO(PREC(P),READ,TO(PI),PI,ADD(PI),TO(PI),SL(10),POWER(-P),
     1 TO(EPS))
1     CALL MLO(X,MODULO(PI),SQ,TO(F),SL(0),TO(Y),SL(1),TO(T))
      N = -1
2     N = N+2
      IF(MLT(Y,ADD(T),TO(Y),T,MULT(F),DIV(SL(-N*(N+1))),TO(T),POS,SUB(
     1 EPS))) ,,2
      RETURN
      END
```

```
      SUBROUTINE MLCREM(I,LP,N,IANS,P)

C  PURPOSE.  SOLVES SYSTEM OF CONGRUENCES IANS≡I(L) (MOD LP(L)),L = 1,N
C  ARGUMENTS.  I,LP INTEGER ARRAYS. IANS ML-VARIABLE.  ELEMENTS OF LP MUST BE
C              RELATIVELY PRIME IN PAIRS
C  RESTRICTIONS.  2 ≤ N ≤ 50,  R+(LP(L)),L = 1,N
C  CALLS  IZI,IZMR,ML
C  B-REGS.  0 - 9
C       FORMS PROD = ∏(L=1,N) LP(L) AND PROD/LP(I) ONCE AND FOR ALL.
C       P MUST BE LARGE ENOUGH TO ENABLE N*PROD*MAX(LP(L)) TO BE CALCULATED
C       EXACTLY.  IANS COMPUTED MODULO PROD -. IF GREATER THAN PROD/2 IANS MADE
C       NEGATIVE.

      DIMENSION I(1),LP(1),A(50)
      CALL MLO(PREC(P),SL(0),TO(IANS))
      IE(III) 4,,4
      III = 1
      CALL MLD(PREC(P),NAMES(PROD,PRODH))
      DO 6 LL = 1,N
6     CALL MLD(PREC(P),NAMES(A(LL)))
      CALL MLO(PREC(P),SL(1),TO(PROD))
      DO 1 L = 1,N
1     CALL MLO(PROD,MULT(SL(LP(L))),TO(PROD))
      CALL MLO(PROD,DIV(SL(2)),TO(PRODH))
      DO 2 L = 1,N
      LLP = LP(L)
      J = 1
      DO 3 LL = 1,N
      LLLP = LP(LL)
      IE(LLLP - LLP) ,3,
      J = IZMR (LLP,0,J,LLLP)
3     CONTINUE
2     CALL MLO(PROD,DIV(SL(LLP)),MULT(SL(IZI(J,LLP))),TO(A(L)))
4     DO 5 L = 1,N
5     CALL MLO(A(L),MULT(SL(I(L))),ADD(IANS),TO(IANS))
      IF  (MLT(IANS,MODULO(PROD),TO(IANS),SUB(PRODH))) 7,7,
      CALL MLO(IANS,SUB(PROD),TO(IANS))
7     RETURN
      END
```

```
      SUBROUTINE MLCVS (X,I,J,K,LOUT,LTEST,N,P)

C  PURPOSE.  ML VERSION OF CONFRA
C  ARGUMENTS.  X ML-VARIABLE,  I,J,K ML-ARRAYS.  P PRECISION.
C              ON RETURN I CONTAINS PARTIAL QUOTIENTS, J,K NUMERATOR,
C              DENOMINATOR OF CONVERGENTS TO X.  PRINTS THESE. COMPUTES UNTIL
C              NUMERATOR * DENOMINATOR >=10**N OR UNTIL LTEST PARTIAL QUOTIENTS
C              HAVE BEEN COMPUTED. WHEN ASTERISKS ARE PRINTED.  LOUT = NUMBER
C              OF CONVERGENTS COMPUTED
C  CALLS  ML
C  B-REGS.  0 = 10

      EXTERNAL RCP,IPART,POS
      DIMENSION I(1),J(1),K(1)
      IF(II) 1,,1
      II = 1
      CALL MLO(PREC(P),NAMES(Y,Z,LIMIT,RECIP))
      CALL MLO(PREC(P),SL(10),POWER(N),TO(LIMIT),LIMIT,RCP,TO(RECIP))
1     CALL MLO(PREC(P),SL(0),TO(J(1)),TO(K(2)),SL(1),TO(J(2)),TO(K(1)))
      CALL MLO(X,POS,TO(Y))
      CALL MLO(PREC(P),NL(2),TEXT(14HCONVERGENTS OF),X,PRINT(10,P))
      L = 3
5     CALL MLO(Y,IPART,TO(I(L)))
      CALL MLO(I(L),MULT(J(L-1)),ADD(J(L-2)),TO(J(L)))
      CALL MLO(I(L),MULT(K(L-1)),ADD(K(L-2)),TO(K(L)))
      CALL MLO(PREC(P),NL(1),SL(L-3),PRINT(5,0),I(L),PRINT(15,0),J(L),
     1 PRINT(40,0),K(L),PRINT(40,0))
      IF(MLT(J(L),ADD(K(L)),SUB(LIMIT))) 4,2,2
4     CALL MLO(Y,SUB(I(L)),TO(Z))
      IF(MLT(Z,SUB(RECIP))) 2,3,3
2     LOUT = L
      RETURN
3     CALL MLO(Z,RCP,TO(Y))
      L = L + 1
      IF(L-LTEST) 5,5,
      CALL MLO(PREC(P),NL(1),TEXT(10H**********))
      LOUT = L
      RETURN
      END
```

```
      SUBROUTINE MLDC(A,N,X,P)

C  PURPOSE,  COMPUTES DISCRIMINANT OF POLYNOMIAL WITH MULTI-LENGTH COEFFICIENTS
C  ARGUMENTS,  A IS ML-ARRAY,  X  ML-VARIABLE, P PRECISION.  X SET TO
C              DISCRIMINANT OF A(1) X**N + ... + A(N+1)
C  RESTRICTIONS,  N < 26 AND SAME FOR ALL CALLS IN ONE RUN.
C  CALLS MLDET,ML
C  B-REGS,  0 - 9

      INTEGER P
      DIMENSION B(49,49),BB(49,49),A(1),AA0(26),AA(26),AA1(26)
      IF(III) 1,,1
    1  III = 1
      N1 = N + 1
      N2 = 2*N - 1
      DO 5 I = 2-N,N2 + 1
      CALL MLD(PREC(P),NAMES(AA(I)))
    5 CALL MLO(PREC(P),SL(0),TO(AA(I)))
      DO 51 I = 1,N2
      DO 50 J = 1,N2
   50 CALL MLD(PREC(P),NAMES(B(I,J),BB(I,J)))
   51 CONTINUE
    1 DO 2 I = 1,N1
    2 CALL MLO(A(I),TO(AA(I)))
      K = 1
      DO 4 I = 2,N2,2
      K = K - 1
      DO 6 J = 1,N2
    6 CALL MLO(AA(J+K),TO(B(I,J)))
    4 CONTINUE
      CALL MLO(PREC(P),SL(0),TO(AA(N1)))
      DO 7 I = 1,N-1
    7 CALL MLO(AA(I),MULT(SL(N1-I)),TO(AA(I)))
      K = 1
      DO 8 I = 1,N2,2
      K = K - 1
      DO 9 J = 1,N2
    9 CALL MLO(AA(J+K),TO(B(I,J)))
    8 CONTINUE
      CALL MLO(PREC(P),SL(N),TO(B(1,1)),SL(1),TO(B(2,1)))
      CALL MLDET(B,BB,X,N2,P,49)
      RETURN
      END
```

```
      SUBROUTINE MLDET(B,A,DET,M,P,IA)

C  PURPOSE.  ML VERSION OF MB02AM.
C  ARGUMENTS.  B,A ML-ARRAYS.  DET ML-VARIABLE.  P PRECISION.  DET SET TO
C              DETERMINANT OF M BY M MATRIX B, WHICH IS COPIED INTO A. IA IS
C              FIRST DIMENSION OF A OR B IN CALLING PROGRAMME.
C  RESTRICTIONS.  M <= 20
C  CALLS  ML
C  B-REGS.  0 - 12

      EXTERNAL COMP,POS
      INTEGER P,DD
      DIMENSION B(IA,1),A(IA,1),D(20),IND(20),JND(20)
      IF(III) 1,,1
      III = 1
      CALL MLD(PREC(P),NAMES(AMAX,STO))
      DO 101 I = 1,20
 101  CALL MLD(PREC(P),NAMES(D(I)))
 1    CALL MLO(PREC(P),SL(0),TO(AMAX))
      DO 2 I = 1,M
      IND(I) = I
      JND(I) = I
      DO 7 J = 1,M
      CALL MLO(B(I,J),TO(A(I,J)))
      IF(MLT(A(I,J),POS,SUB(AMAX))) 7,7,3
 3    CALL MLO(A(I,J),POS,TO(AMAX))
      I4 = I
      J4 = J
 7    CONTINUE
 2    CONTINUE
      DD = 1
      MM = M-1
      DO 111 J = 1,MM
      IF(I4-J) 6,6,4
 4    DD = -DD
      ISTO = IND(J)
      IND(J) = IND(I4)
      IND(I4) = ISTO
      DO 5 K = 1,M
 5    CALL MLO(A(I4,K),TO(STO),A(J,K),TO(A(I4,K)),STO,TO(A(J,K)))
 6    IF(J4-J) 8,8,9
 9    DD = -DD
      ISTO = JND(J)
      JND(J) = JND(J4)
      JND(J4) = ISTO
      DO 12 K = 1,M
 12   CALL MLO(A(K,J4),TO(STO),A(K,J),TO(A(K,J4)),STO,TO(A(K,J)))
 8    CALL MLO(PREC(P),SL(0),TO(AMAX))
      J1 = J + 1
      DO 11 I = J1,M
```

```
      CALL MLO(A(I,J),DIV(A(J,J)),COMP,TO(STO))
      DO 10 K = 1,M
      CALL MLO(A(J,K),MULT(STO),ADD(A(I,K)),TO(A(I,K)))
      IF (K-J) 10,10,15
15    IF(MLT(A(I,K),POS,SUB(AMAX))) 10,10,17
17    CALL MLO(A(I,K),POS,TO(AMAX))
      I4 = I
      J4 = K
10    CONTINUE
      CALL MLO(STO,TO(A(I,J)))
11    CONTINUE
111   CONTINUE
      CALL MLO(PREC(P),SL(DD),TO(D(1)))
      DO 18 I = 1,MM
18    CALL MLO(A(I,I),MULT(D(I)),TO(D(I + 1)))
      CALL MLO(A(M,M),MULT(D(M)),TO(DET))
      RETURN
      END
```

```
      SUBROUTINE MLEXP(X,Y,P)

C  PURPOSE:  COMPUTES ML EXPONENTIAL
C  ARGUMENTS.  X,Y ML-VARIABLES.  P PRECISION.  SETS Y = EXP(X)
C  CALLS  ML
C  B-REGS.  0 - 4
C        COMPUTES E = EXP(1) ONCE AND FOR ALL ON FIRST EXECUTION.

      EXTERNAL IPART,FPART,POS
      IF(II) 2,,2
      II = 1
      CALL MLD(PREC(P),NAMES(EPS,T,F,E))
      CALL MLO(PREC(P),SL(1),TO(E),TO(F),SL(10),POWER(-P),TO(EPS))
      N = 1
3     N = N + 1
      IF(MLT(E,ADD(F),TO(E),F,DIV(SL(N)),TO(F),SUB(EPS))) ,,3
2     CALL MLO(X,IPART,TOSL(I),X,FPART,TO(F),TO(T),SL(1),TO(Y))
      N = 1
1     N = N + 1
      IF(MLT(Y,ADD(T),TO(Y),T,MULT(F),DIV(SL(N)),TO(T),POS,SUB(EPS))),,1
      CALL MLO(E,POWER(I),MULT(Y),TO(Y))
      RETURN
      END
```

```
      SUBROUTINE MLEXPO(X,Y,Z,P)

C  PURPOSE,  COMPUTES ML EXPONENTIAL
C  ARGUMENTS,  X,Y,Z  ML-VARIABLES,  P PRECISION,  SETS Z = X**Y
C  CALLS  MLEXP,MLLOG,ML
C  B-REGS,  0 - 5

      IF(III) 1,,1
      III = 1
      CALL MLD(PREC(P),NAMES(A))
1     CALL MLLOG(X,A,P)
      CALL MLD(Y,MULT(A),TO(A))
      CALL MLEXP(A,Z,P)
      RETURN
      END
```

```
      SUBROUTINE MLJ(C,D,AA,BB,N,NN,NSPEC,P)

C  PURPOSE,  COMPUTES KLEIN'S INVARIANT J (SMALL J)
C  ARGUMENTS,  C,D,AA,BB ARE ML-VARIABLES.  P PRECISION.
C              IF NSPEC NEQ 0, C AND NN ARE IGNORED AND AA IS SET TO
C              RE J(1/2 + D.I),  BB TO IM J(1/2 + D.I), USING N TERMS OF J =
C              1, 744, 196884, ... (MLCOS NOT ENTERED).
C              IF NSPEC = 0,  AA IS SET TO RE J(C + D.I),  BB TO IM J(C + D.I),
C              USING N TERMS OF EXPANSION.  IF BOTH C AND D ARE OF FORM
C              C'/INT,  D'/INT, WHERE INT IS AN EVEN INTEGER, SET NN = INT
C              OTHERWISE SET NN = 0.
C  RESTRICTIONS,  3≤N≤400.
C  CALLS  MLCOS,MLEXP,ML
C  B-REGS,  0 - 11
C        ON FIRST EXECUTION READS P FROM CURRENT INPUT STREAM AND READS
C        C(1) = 1,  C(2) = 744, ..., C(N) FROM INPUT STREAM 1, SELECTING INPUT
C        STREAM 0 AFTERWARDS

      DIMENSION AJ(400)
      INTEGER P
      EXTERNAL READ,RCP,SQROOT,SQ,COMP,POS
      IF(III) 1,,1
      III = 1
      PP = 2.0*3.14159265359
      CALL MLD(PREC(P),NAMES(PI,X,Y,Z,W,V,A,B,EPS,S,SS))
      CALL MLO(PREC(P),READ,TO(PI),PI,ADD(PI),TO(PI),SL(10),POWER(-P),
     1 TO(EPS))
      CALL INPSEL(1)
      DO 2 I = 1,N
      CALL MLD(PREC(P),NAMES(AJ(I)))
2     CALL MLO(PREC(P),READ,TO(AJ(I)))
      CALL INPSEL(0)
1     IF(NSPEC) ,9,
      CALL MLO(D,MULT(PI),TO(A))
      CALL MLEXP(A,S,P)
      CALL MLO(S,COMP,TO(B),RCP,TO(A),B,ADD(AJ(2)),TO(S),SL(1),TO(B))
      DO 10 I = 3,N
      IF(MLT(B,MULT(A),TO(B),MULT(AJ(I)),TO(SS),ADD(S),TO(S),SS,POS,
     1 SUB(EPS))) 11,11,
10    CONTINUE
11    CALL MLO(S,TO(AA),SL(0),TO(BB))
      RETURN
9     CALL MLO(C,TOSL(A1),MULT(PI),TO(A),COMP,TO(Z),D,TOSL(B1),MULT(PI),
     1 DO(B))
      Z1 = PP*A1
      Z2 = -Z1
      NNN = NN/2
      CALL MLEXP(B,X,P)
      CALL MLO(X,RCP,TO(Y))
      CALL MLCOS(Z,W,P)
```

```
      CALL MLO(AJ(1),MULT(X),TO(V),MULT(W),TO(AA))
      CALL MLO(W,SQ,COMP,ADD(SL(1)),SQROOT,TO(W))
      IF(SINF(Z2)) ,3,3
      CALL MLO(W,COMP,TO(W))
3     CALL MLO(V,MULT(W),TO(BB))
      K = 0
      J = -1
      DO 4 I = 2,N
      J = J * -1
      CALL MLO(X,MULT(Y),TO(X),Z,ADD(A),TO(Z))
      Z2 = Z2 * Z1
6     CALL MLCOS(Z,W,P)
7     CALL MLO(AJ(I),MULT(X),TO(V),MULT(W),ADD(AA),TO(AA))
      IF(J-K)  8,,8
      K = K* NNN
      GO TO 4
8     CALL MLO(W,SQ,COMP,ADD(SL(1)),SQROOT,TO(W))
      IF(SINF(Z2))  ,5,5
      CALL MLO(W,COMP,TO(W))
5     CALL MLO(V,MULT(W),ADD(BB),TO(BB))
4     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MLLOG(X,Y,P)

C  PURPOSE,  COMPUTES ML LOGARITHM.
C  ARGUMENTS.  X,Y ML-VARIABLES.  P PRECISION.  Y SET TO LOG(X)
C  CALLS  MLEXP,ML
C  B-REGS.  0 - 4

      EXTERNAL POS
      INTEGER P
      IF(II) 2,,2
      II = 1
      CALL MLD(PREC(P),NAMES(Z,A,B,C,EPS))
      CALL MLO(PREC(P),SL(10),POWER(-P),TO(EPS))
2     CALL MLO(X,TOSL(XX))
      XX = LOGF(XX)
      CALL MLO(PREC(P),SL(XX),TO(Y))
      CALL MLEXP(Y,Z,P)
      CALL MLO(X,DIV(Z),SUB(SL(1)),TO(Z),TO(B),TO(A))
      N = 1
1     N = N + 1
      CALL MLO(A,MULT(B),TO(A),DIV(SL(-N)),ADD(Z),TO(Z))
      N = N + 1
      CALL MLO(A,MULT(B),TO(A),DIV(SL(N)),TO(C),ADD(Z),TO(Z))
      IF(MLT(C,POS,SUB(EPS))) ,,1
      CALL MLO(Y,ADD(Z),TO(Y))
      RETURN
      END
```

```
      SUBROUTINE MLPOW(A,N1,N2,Y,XX,P)

C  PURPOSE,  ML EXPONENTIATION WITH RATIONAL EXPONENT, USING NEWTON ITERATION.
C  ARGUMENTS,  A,Y,XX ML-VARIABLES,  SETS Y = A**(N1/N2),  CALCULATED WITH
C              PRECISION P.  XX MUST INITIALLY CONTAIN APPROXIMATION TO Y.
C  CALLS  ML
C  B-REGS,  0 - 7

      EXTERNAL RCP,POS
      INTEGER P
      IF(III) 1,,1
      III = 1
      CALL MLD(PREC(P),NAMES(EPS,B,R,X,XXX))
      CALL MLO(PREC(P),SL(10),POWER(-P+2),TO(EPS))
1     NN = N2-1
      CALL MLO(A,POWER(N1),DIV(SL(N2)),TO(B),XX,TO(X),SL(NN),DIV(SL(N2))
     1 ,TO(R))
2     CALL MLO(X,TO(XXX))
      CALL MLO(X,POWER(NN),RCP,MULT(B),ADD(X,MULT(R)),TO(X))
      IF(MLT(X,SUB(XXX),POS,SUB(EPS))) ,,2
      CALL MLO(X,TO(Y))
      RETURN
      END
```

```
      SUBROUTINE MLPOW3(A,Y,XX,P)

C  PURPOSE.  COMPUTES CUBE ROOT USING NEWTON ITERATION.
C  ARGUMENTS.  A,Y  ML-VARIABLES.  P PRECISION.  SETS Y = A**(1/3).
C              XX MUST INITIALLY CONTAIN A SINGLE-LENGTH APPROXIMATION TO Y.
C  CALLS  ML
C  B-REGS,  0 - 5

      EXTERNAL RCP,POS,SQ
      INTEGER P
      IF(III) 1,,1
      III = 1
      CALL MLD(PREC(P),NAMES(EPS,B,R,X,XXX))
      CALL MLO(PREC(P),SL(10),POWER(-P*3),TO(EPS))
1     CALL MLO(A,DIV(SL(3)),TO(B),SL(XX),TO(X),SL(2),DIV(SL(3)),TO(R))
2     CALL MLO(X,TO(XXX))
      CALL MLO(X,SQ,RCP,MULT(B),ADD(X,MULT(R)),TO(X))
      IF(MLT(X,SUB(XXX),POS,SUB(EPS))) ,,2
      CALL MLO(X,TO(Y))
      RETURN
      END
```

```
      SUBROUTINE MLROOT(AA,NN,XINIT,ROOT,P)

C  PURPOSE.  COMPUTES ROOT OF ALGEBRAIC EQUATION.
C   ARGUMENTS.  AA ML-ARRAY.  ROOT ML-VARIABLE.  EQUATION IS AA(1) X**NN + ...
C               +AA(NN+1).  XINIT IS SINGLE LENGTH APPROXIMATION TO ROOT
C               P IS PRECISION.
C  RESTRICTIONS.  NN < 50
C  CALLS ML
C  B-REGS.  0 - 8

      DIMENSION AA(1),A(50)
      INTEGER P,POWR
      EXTERNAL COMP
      IF(III) 1,,1
      III = 1
      CALL MLD(PREC(P),NAMES(X,Y,F,G,DEL))
1     POWR = 0.4343*LOGF(ABSF(XINIT)+1)-P
      N = NN
      DO 2 I = 1,N + 1
2     A(I) = AA(I)
      CALL MLO(PREC(P),SL(XINIT),TO(X),SL(10),POWER(POWR),TO(DEL))
      CALL MLO(PREC(P),SL(A(1)),TO(F))
      DO 3 I = 2,N+1
3     CALL MLO(F,MULT(X),ADD(SL(A(I))),TO(F))
100   CALL MLO(X,ADD(DEL),TO(Y))
      CALL MLO(PREC(P),SL(A(1)),TO(G))
      DO 4 I = 2,N+1
4     CALL MLO(G,MULT(Y),ADD(SL(A(I))),TO(G))
      IF(MLT(F)) 102,199,101
101   IF(MLT(G)) 199,199,103
102   IF(MLT(G)) 104,199,199
103   IF(MLT(F,SUB(G))) 106,204,107
104   IF(MLT(G,SUB(F))) 106,204,107
106   CALL MLO(DEL,COMP,TO(DEL))
107   CALL MLO(DEL,MULT(F),DIV(F,SUB(G)),ADD(X),TO(X))
      CALL MLO(PREC(P),SL(A(1)),TO(F))
      DO 5 I = 2,N+1
5     CALL MLO(F,MULT(X),ADD(SL(A(I))),TO(F))
      IF(MLT(F)) 109,100,
      IF(MLT(G)) 111,204,100
109   IF(MLT(G)) 100,204,111
111   CALL MLO(DEL,COMP,TO(DEL))
      GO TO 100
199   CALL MLO(X,TO(ROOT))
      RETURN
204   CALL MLO(PREC(P),SL(XINIT),TO(ROOT),NL(1),TEXT(21H*** NO ROOT FOUN
     1D ***))
      RETURN
      END
```

```
      SUBROUTINE MLSIN(X,Y,P)

C  PURPOSE,  COMPUTES ML SINE FUNCTION
C  ARGUMENTS,  X,Y  ML-VARIABLES,  P IS PRECISION.  SETS Y = SIN(X)
C  CALLS  ML
C  B-REGS,  0 - 4
C         READS # FROM CURRENT INPUT STREAM ON FIRST EXECUTION.

      INTEGER P
      EXTERNAL READ,SQ,POS
      IF(III)1,,1
      III = 1
      CALL MLD(PREC(P),NAMES(EPS,PI,F,T))
      CALL MLO(PREC(P),READ,TO(PI),PI,ADD(PI),TO(PI),SL(10),POWER(-P),
     1 TO(EPS))
1     CALL MLO(X,MODULO(PI),TO(T),SQ,TO(F),SL(0),TO(Y))
      N= 0
2     N = N+2
      IF(MLT(Y,ADD(T),TO(Y),T,MULT(F),DIV(SL(-N*(N+1))),TO(T),POS,SUB(
     1 EPS))) ,,2
      RETURN
      END
```