Some applications of computing to number theory

Muir, S. T. E.

How to cite:

Use policy
The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

• a full bibliographic reference is made to the original source
• a link is made to the metadata record in Durham E-Theses
• the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders. Please consult the full Durham E-Theses policy for further details.
SOME APPLICATIONS OF COMPUTING TO NUMBER THEORY

by

S. T. E. MUIR

CONTENTS

Chapter 1     ... ... ... ... ... ... ... Page 1
Chapter 2     ... ... ... ... ... ... ... Page 9
Chapter 3     ... ... ... ... ... ... ... Page 24
Chapter 4     ... ... ... ... ... ... ... Page 58
References    ... ... ... ... ... ... ... Page 65
Computer output
ABSTRACT

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 \( j\left(\frac{-1+i\sqrt{2347}}{2}\right) \) 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 \( \pi \) 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.

1.5. 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 \( \mu \text{sec} \). per word, where \( 1 \mu \text{sec.} = 10^{-6} \) 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 16 1/2 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: 2μsecs. for floating-point addition, approximately 6μsecs. for floating-point multiplication, and from 1.6 to 1.8μsecs. 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\(\tau\)-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

\[ \tau = \frac{ax + b}{cx + d} \]  \hspace{1cm} (2.2.1)

where \(a, b, c\) and \(d\) are rational integers with \(ad - bc = 1\). This group, denoted
by $\Gamma$, is generated by the transformations $S$ and $T$ where
\[ ST = T + 1 \quad T = -\frac{1}{\tau} \]  \hspace{1cm} (2.2.2)
and is the free product of the cyclic groups $\{T\}$ and $\{P\}$ of order 2 and 3 respectively, where
\[ P = TS \quad Pt = -\frac{1}{\tau+1} \quad T^2 = P^3 = 1 \, . \]

It is a discontinuous group whose fundamental domain $\mathcal{F}$ is shown as the shaded region in the figure in 3.6. The vertical sides of the boundary of $\mathcal{F}$ 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 $\Gamma$ (see Klein - Fricke (11), p.591), is denoted by $j$ where
\[ j(\frac{a\tau + b}{c\tau + d}) = j(\tau) \quad ad - bc = 1 \, . \]

$j$ is zero at $\tau = \rho = e^{i\frac{2\pi}{3}}$, 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 $x$
\[ j = \sum_{n=1}^{\infty} c(n)x^n = x^{-1} + 744 + 196884x + ... \]  \hspace{1cm} (2.2.3)
where the coefficients $c(n)$ are positive integers. These coefficients enjoy certain congruence properties. For example

if \[ n \equiv 0 \pmod{2^{a+3b}5^c7^d11^e} \]
then \[ c(n) \equiv 0 \pmod{2^{3a+8b+3c+17^d11^e}} \, . \]

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'.

(10)
If \( \forall \tau = \frac{a\tau + b'}{c\tau + d'} \in \Gamma \), we write \( V \equiv V'(\mod a) \) if and only if
\[ a \equiv a', \quad b \equiv b', \quad c \equiv c', \quad d \equiv d'(\mod n). \]
An important subgroup of \( \Gamma \) is the
set of all transformations

\[ V = \tau I \pmod{n} \]

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 \( \Gamma \). By taking \( n = 1 \), \( \Gamma \) may be written as \( \Gamma(1) \).

A subgroup \( G \) of finite index \( \mu \) in \( \Gamma \) has a fundamental domain consisting of \( \mu \) copies of \( F \). If the elements of \( G \) conjugate in \( \Gamma \) 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 \( \Gamma \) to a non-zero power of \( S \) be conjugate in \( G \) to some power of one of

\[ S^{\mu_1}, S_2^{\mu_2} S_2^{-1}, \ldots, S_t^{\mu_t} S_t^{-1} \]

where \( g_t = I, g_2, \ldots, g_t \) are in \( \Gamma \) and \( g_1 g_j^{-1} \notin G \). 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 \( \Gamma \). We have

\[ \mu = \mu_1 + \mu_2 + \cdots + \mu_t \]

and the genus \( g \) of \( G \) is given by

\[ g = 1 + \frac{\mu}{12} - \frac{t}{2} - \frac{e_2}{4} - \frac{e_3}{3} \]  

Almost all subgroups of \( \Gamma \) 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 \( \Gamma \) then \( f(\tau) \) is a modular form on \( G \) if

\[ f(\tau) \]
(1) \( f(\tau) \) is single-valued and regular except for poles at the cusps of \( \mathcal{G} \).

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

When \( \omega = 0 \), \( f \) is called a modular function on \( \mathcal{G} \); when \( \omega = 1 \), \( f \) is called a differential on \( \mathcal{G} \). 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. \( \Gamma_q \): 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 \( \Gamma \) of genus 0 can be completely specified by giving:

(1) a specification for the group, i.e. a set of integers \( \mu \geq 1, t \geq 1, e_2 \geq 0, e_3 \geq 0, \mu_i \geq 1 \) and, if \( h > 1 \), a set \( (a_i, v_i) \) with \( a_i \geq 1, v_i > v_{i+1} > \ldots > v_s \), for \( i = 1 \) to \( s \), satisfying

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

(3) a relation between the coefficients of the polynomials in \( \zeta \) derivable from the zero constant term in the expansion of \( \zeta \) at \( \infty \).

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 \( \Gamma_q \), where \( t = 1, \mu = \mu_1 = 9, g = 1, e = 1 \) and \( e = 0 \), the elimination of \( j \) from these equations produces \( 2\mu = 18 \) simultaneous non-linear equations between as many unknowns whose solution effectively specifies \( \Gamma_q \). 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:

\[(x^3 + x^2 y + x y^2)(x^2 + x y + y^2) = (x^4 + x^3 y + x^2 y^2 + x y^3 + y^4)\]  
\[(x^3 + x^2 y + x y^2)(x^2 + x y + y^2) = (x^4 + x^3 y + x^2 y^2 + x y^3 + y^4)\]

\[(x^3 + x_2 + x_3)(x^2 + x_5 + x_6)^2 - (x^4 + x_8 + x_9 + x_{10} + x_{11})^2\]
\[= (x^3 + x_2 + x_3)(x^2 + x_5 + x_6)^2 - (x^4 + x_8 + x_9 + x_{10} + x_{11} - 1728)^2\]
\[= x(x^4 + x_5 + x_6 + x_7 + x_8)^2\]

(2.3.1)

Thus the 18 equations are

\[x_3 x_6 - x_11 = x_14^3\]  \[\text{(coefficients of } x^0 \text{ in (2.3.1))}\]
\[x_2 x_6 + 2x_7 + x_6 - 2x_{10} (x_{11} - 1728) = x_{18}^2\]  \[\text{(coefficients of } x^1 \text{ in (2.3.1))}\]
\[x_1 + 2x_4 - x_7 = 2x_{15}\]  \[\text{(coefficients of } x^8 \text{ in (2.3.1))}\]

(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 \( f_1 = x_3 x_2^2 - x_5^2 x_2 - x_3^2 x_4^2 \), \( f_6 = x_1^2 + 2 x_1 x_7^2 - 2 x_7^2 - 2 x_1 x_7^2 \).

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 \( \frac{m}{n} \cdot k^n \), where

\[ k = \left( \frac{3^3}{2^8} \right)^{1/4} = 0.7788578471 \ldots; \]

the last column is an approximation to the solution.

(15)
<table>
<thead>
<tr>
<th>(i)</th>
<th>(m)</th>
<th>(a)</th>
<th>(x_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>225/4</td>
<td>2</td>
<td>34.12234</td>
</tr>
<tr>
<td>2</td>
<td>960</td>
<td>4</td>
<td>353.26778</td>
</tr>
<tr>
<td>3</td>
<td>4096</td>
<td>6</td>
<td>914.34500</td>
</tr>
<tr>
<td>4</td>
<td>54</td>
<td>2</td>
<td>32.75745</td>
</tr>
<tr>
<td>5</td>
<td>933</td>
<td>4</td>
<td>343.33212</td>
</tr>
<tr>
<td>6</td>
<td>5041</td>
<td>6</td>
<td>1125.29372</td>
</tr>
<tr>
<td>7</td>
<td>-9/2</td>
<td>1</td>
<td>-3.50486</td>
</tr>
<tr>
<td>8</td>
<td>-321</td>
<td>3</td>
<td>-151.66299</td>
</tr>
<tr>
<td>9</td>
<td>-16425/2</td>
<td>5</td>
<td>2353.78278</td>
</tr>
<tr>
<td>10</td>
<td>-174303/2</td>
<td>7</td>
<td>-15152.42927</td>
</tr>
<tr>
<td>11</td>
<td>-306240</td>
<td>9</td>
<td>-32298.75000</td>
</tr>
<tr>
<td>12</td>
<td>48</td>
<td>2</td>
<td>29.11775</td>
</tr>
<tr>
<td>13</td>
<td>672</td>
<td>4</td>
<td>247.28744</td>
</tr>
<tr>
<td>14</td>
<td>2176</td>
<td>6</td>
<td>485.74472</td>
</tr>
<tr>
<td>15</td>
<td>72</td>
<td>2</td>
<td>43.67660</td>
</tr>
<tr>
<td>16</td>
<td>1872</td>
<td>4</td>
<td>688.87217</td>
</tr>
<tr>
<td>17</td>
<td>20544</td>
<td>6</td>
<td>4586.00164</td>
</tr>
<tr>
<td>18</td>
<td>81792</td>
<td>8</td>
<td>11075.83371</td>
</tr>
</tbody>
</table>

These solutions were obtained in the following way. On \(\Gamma_9\) we have two functions

\[
x = \xi^{-2} + \sum_{n=1}^{\infty} a_1(n) \xi^n
\]

and

\[
y = \xi^{-3} + \sum_{n=1}^{\infty} a_2(n) \xi^n
\]

where

\[
\xi = e^{\frac{2 \pi i}{\varphi}}
\]

Every function on \(\Gamma_9\) is an algebraic function of \(x\) and \(y\). Normally the coefficients \(a_1(n)\) and \(a_2(n)\) are complex; in the case of \(\Gamma_9\), 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

\[
\begin{align*}
\tau_1 &= b_1 + e^{i \theta} \\
\tau_2 &= c_1 + e^{i (\pi - \theta)}
\end{align*}
\]

then \(x(\tau_1) = x(\tau_2)\) and \(y(\tau_1) = y(\tau_2)\) where \(\frac{\pi}{2} \leq \theta \leq \frac{3\pi}{2}\) and the \(b_1, c_1\) are suitably chosen pairs of integers. On \(\Gamma_9\) we have, for example, the pairings

\[
b_1 = 0, c_1 = 0 ; b_2 = 1, c_2 = 4 ; b_3 = 2, c_3 = 6\text{ (amongst others)}.
\]

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 \( \Theta \) 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 \( y = y_{\text{old}} + \frac{3}{2}x \) and by fixing the constant terms in the expansions of \( x \) and \( y \), we obtain the relations

\[
\begin{align*}
y^2 &= x^3 + 225/4k^2x^2 + 960k^4x + 4096k^6 \\
j &= y(x^3 + 54k^2x^2 + 933k^4x + 5041k^6) - \frac{1}{4}(9kxx^4 + 642k^3x^3 + 1645k^5x^2 + 17430k^7x + 612480k^9)
\end{align*}
\]

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 \( \Gamma_9 \), 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 \( \frac{m}{\sqrt{3}^n} \), where \( \sqrt{3} \) \( m \) (except for \( i=3,6 \) and multiples of 9). The right-hand entries against the primes \( p \) are the character sums \( \sum_{x=0}^{p-1} \left( \frac{y^2}{p} \right) \).

(17)
<table>
<thead>
<tr>
<th>n</th>
<th>n</th>
<th>i</th>
<th>\text{character sum}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>-3</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>-5</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>-5</td>
<td>0</td>
<td>7</td>
<td>-2</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>-875</td>
<td>4</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>274</td>
<td>4</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>539</td>
<td>2</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>3932</td>
<td>4</td>
<td>13</td>
<td>-2</td>
</tr>
<tr>
<td>9365</td>
<td>4</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>-910</td>
<td>2</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>10159</td>
<td>4</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>556</td>
<td>4</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>-17</td>
<td>13787</td>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>37</td>
<td>94725</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>75850</td>
<td>5</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>-57</td>
<td>10514</td>
<td>8</td>
<td>22</td>
</tr>
<tr>
<td>16</td>
<td>35149</td>
<td>8</td>
<td>23</td>
</tr>
<tr>
<td>2</td>
<td>50073</td>
<td>5</td>
<td>24</td>
</tr>
<tr>
<td>-17680</td>
<td>8</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>46244</td>
<td>8</td>
<td>26</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>-13695</td>
<td>44161</td>
<td>13</td>
<td>28</td>
</tr>
<tr>
<td>-37955</td>
<td>93836</td>
<td>13</td>
<td>29</td>
</tr>
<tr>
<td>-14</td>
<td>98630</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>59133</td>
<td>93124</td>
<td>13</td>
<td>31</td>
</tr>
<tr>
<td>-1</td>
<td>98424</td>
<td>52435</td>
<td>15</td>
</tr>
<tr>
<td>-202</td>
<td>63550</td>
<td>8</td>
<td>33</td>
</tr>
<tr>
<td>-1</td>
<td>12065</td>
<td>50788</td>
<td>13</td>
</tr>
<tr>
<td>-1</td>
<td>34310</td>
<td>87490</td>
<td>13</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>-7755</td>
<td>40277</td>
<td>59242</td>
<td>84974</td>
</tr>
<tr>
<td>-62070</td>
<td>08165</td>
<td>13546</td>
<td>23220</td>
</tr>
<tr>
<td>-2</td>
<td>46835</td>
<td>24201</td>
<td>03676</td>
</tr>
<tr>
<td>-27813</td>
<td>30552</td>
<td>38924</td>
<td>86914</td>
</tr>
<tr>
<td>-85568</td>
<td>15354</td>
<td>28903</td>
<td>46806</td>
</tr>
<tr>
<td>-121</td>
<td>93450</td>
<td>57370</td>
<td>27048</td>
</tr>
<tr>
<td>12140</td>
<td>91762</td>
<td>41076</td>
<td>27279</td>
</tr>
<tr>
<td>-1</td>
<td>94862</td>
<td>13591</td>
<td>18374</td>
</tr>
</tbody>
</table>

(18)
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 $k^\mu$, 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 $\mu$ where
\begin{enumerate}
  \item if $\mu$ is even, $N = \frac{1}{2}(\mu-4)$ and $e(N+3) = 1$
  \item if $\mu$ is odd, $N = \frac{1}{2}(\mu-3)$, $d(N+1) = 1$, and $e(N+3) = 0$.
\end{enumerate}
Thus, in the case of $\Gamma_9$, $\mu=9$ and $N=3$, with the relations (2.3.6) input to the subroutine. We may eliminate $k$ from these relations using the transformations $x_{\text{new}} = x/k^2$, $y_{\text{new}} = y/k^3$ 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 $\alpha$ in the expansion of $x$ and $\beta$ in the expansion of $y$ are being determined as solutions of two simultaneous linear equations in the unknowns $\alpha$ and $\beta$, obtained by equating the relevant powers of $\xi$ 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 $2, \mu$ and $k^\mu$. 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 $f_x$, 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 $f_x$ is that, for $p$ prime,

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

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

$$y^2 = x^3 + x_1 x^2 + x_1 x + x_0$$

and obtain $\frac{dx}{y}$ with coefficients $a(n)$, say. Let $c(p) = \sum_{x=0}^{p-1} \left( \frac{x^2}{p} \right)$ and let $\Delta$ be the discriminant of the cubic $x^3 + x_1 x^2 + x_1 x + x_0$. 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 $p | c(p)$,

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

(20)
where \( \lambda_k \equiv p \pmod{p^k} \);

(3) if \( p | \Delta \) and \( p \nmid c(p) \),
\[
a(p^k) + \lambda_k a(p^{k-1}) \equiv 0 \pmod{p^k}
\]
where \( \lambda_k \equiv -c(p) \pmod{p^k} \);

(4) if \( p | \Delta \) and \( p | c(p) \),
\[
a(p^k) \equiv 0 \pmod{p^k}.
\]

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
\[
f(x) = \prod_{m=1}^{\infty} \left(1 - x^m\right) = 1 - x - x^2 + x^5 + x^7 - \ldots
\]
\[
= 1 + \sum_{n \geq 1} (-1)^n \left\{ \frac{1}{n} \frac{1}{n(5n-1)} + \frac{1}{5n(5n+1)} \right\}
\]
(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) \( x^f_{24}(x) = \sum_{n \geq 1} F_1(n)x^n \), \( F_1(n) = \tau(n) \), Ramanujan's \( \tau \)-function

(2) \( x^f_{12}(x^2) = \sum_{n \geq 1} F_2(n)x^n \), \( F_2(2n) = 0 \)

(3) \( x^f_{4}(x)x^f_{4}(x^5) = \sum_{n \geq 1} F_3(n)x^n \)

(4) \( x^f_{2}(x)x^f_{2}(x^{11}) = \sum_{n \geq 1} F_4(n)x^n \).

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 QEULMUL. 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 $\theta_p^{(n)} = \cos^{-1}(F_n(p)/2p^{k_n/2})$, $0 \leq \theta_p^{(n)} \leq \pi$, 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 $\chi^2$-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 $\Gamma_q$ with

$$-\frac{dx}{q^2} = \sum_{n=1}^{\infty} a(n) x^n$$

where, for example, $a(2) = -k$, $a(3) = -3k^2$, and $q^2 = 27/256$. The sample, however, was too small for a statistical analysis to be performed.
<table>
<thead>
<tr>
<th>p</th>
<th>a(p)</th>
<th>a(p)/2fp</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-0.7785785</td>
<td>-0.27536783</td>
</tr>
<tr>
<td>3</td>
<td>-1.81985864</td>
<td>-0.52534794</td>
</tr>
<tr>
<td>5</td>
<td>-1.8393637</td>
<td>-0.41442228</td>
</tr>
<tr>
<td>7</td>
<td>-1.11614136</td>
<td>-0.21093089</td>
</tr>
<tr>
<td>11</td>
<td>0.27787376</td>
<td>0.04189105</td>
</tr>
<tr>
<td>13</td>
<td>2.4194998</td>
<td>0.33544801</td>
</tr>
<tr>
<td>17</td>
<td>0.12566963</td>
<td>0.01526394</td>
</tr>
<tr>
<td>19</td>
<td>-2.90559048</td>
<td>-0.3329409</td>
</tr>
<tr>
<td>23</td>
<td>1.02015859</td>
<td>0.10655889</td>
</tr>
<tr>
<td>29</td>
<td>-2.17536941</td>
<td>-0.2017798</td>
</tr>
<tr>
<td>31</td>
<td>2.05592107</td>
<td>0.18462716</td>
</tr>
<tr>
<td>37</td>
<td>7.17638578</td>
<td>0.58989520</td>
</tr>
<tr>
<td>41</td>
<td>-1.28208539</td>
<td>-0.10011405</td>
</tr>
<tr>
<td>43</td>
<td>-5.1254484</td>
<td>-0.39081913</td>
</tr>
<tr>
<td>47</td>
<td>1.18831601</td>
<td>0.08666685</td>
</tr>
<tr>
<td>53</td>
<td>5.65045095</td>
<td>0.38807456</td>
</tr>
<tr>
<td>59</td>
<td>2.88925528</td>
<td>0.18807450</td>
</tr>
<tr>
<td>61</td>
<td>11.48031518</td>
<td>0.73495187</td>
</tr>
<tr>
<td>67</td>
<td>-5.97525614</td>
<td>-0.36499686</td>
</tr>
<tr>
<td>71</td>
<td>-2.17313009</td>
<td>-0.12895155</td>
</tr>
<tr>
<td>73</td>
<td>-2.60818903</td>
<td>-0.15263272</td>
</tr>
<tr>
<td>79</td>
<td>0.64310334</td>
<td>0.03617739</td>
</tr>
<tr>
<td>83</td>
<td>3.39551235</td>
<td>0.18580413</td>
</tr>
<tr>
<td>89</td>
<td>-4.15760741</td>
<td>-0.22035275</td>
</tr>
<tr>
<td>97</td>
<td>8.16229765</td>
<td>0.41437788</td>
</tr>
<tr>
<td>101</td>
<td>-10.88634463</td>
<td>-0.54161589</td>
</tr>
<tr>
<td>103</td>
<td>-2.959916217</td>
<td>-0.14578746</td>
</tr>
<tr>
<td>107</td>
<td>-7.62915465</td>
<td>-0.36876911</td>
</tr>
<tr>
<td>109</td>
<td>6.42390350</td>
<td>0.30764918</td>
</tr>
<tr>
<td>113</td>
<td>-0.98011192</td>
<td>-0.04610059</td>
</tr>
<tr>
<td>127</td>
<td>14.86671545</td>
<td>0.65960384</td>
</tr>
<tr>
<td>131</td>
<td>-2.78541510</td>
<td>-0.12168142</td>
</tr>
<tr>
<td>137</td>
<td>-7.79166490</td>
<td>-0.33284343</td>
</tr>
<tr>
<td>139</td>
<td>5.46060717</td>
<td>0.25158133</td>
</tr>
<tr>
<td>149</td>
<td>-13.3630276</td>
<td>-0.54739450</td>
</tr>
<tr>
<td>151</td>
<td>-3.71055400</td>
<td>-0.15098030</td>
</tr>
<tr>
<td>157</td>
<td>10.00373368</td>
<td>0.39919243</td>
</tr>
<tr>
<td>163</td>
<td>-2.89561236</td>
<td>-0.11340876</td>
</tr>
<tr>
<td>167</td>
<td>9.11865499</td>
<td>0.35281135</td>
</tr>
<tr>
<td>173</td>
<td>25.67556582</td>
<td>0.90000997</td>
</tr>
<tr>
<td>179</td>
<td>-1.16126222</td>
<td>-0.04339706</td>
</tr>
<tr>
<td>181</td>
<td>-7.72916906</td>
<td>-0.28725231</td>
</tr>
<tr>
<td>191</td>
<td>-6.72459301</td>
<td>-0.24328724</td>
</tr>
<tr>
<td>193</td>
<td>0.31685666</td>
<td>0.01140392</td>
</tr>
<tr>
<td>197</td>
<td>-8.14854663</td>
<td>-0.29027995</td>
</tr>
<tr>
<td>199</td>
<td>18.27192204</td>
<td>0.64763111</td>
</tr>
</tbody>
</table>

\[(23)\]
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. $3.716 \times 10^3$. 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 net 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.
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 P. Lunn on 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(4). The routines are called from Fortran programmes (or, indeed, ASP programmes) by two subroutines LLO and LIT, whose arguments are functions which operate on multi-length variables. Thus, the Fortran statement

\[ X = \frac{Y + \text{SQRCT}(Z + A \times B)}{C} \]  

(3.2.1)

translates into the statement

\[ \text{CALL LLO (A, LULT(B), ADD(Z), SQRCT, ADD(Y), DIV(C), TO(X))} \]  

(3.2.2)

where A, B, Z, Y, C and X are now multi-length variables and LULT, ADD, SQRCT, 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 LLD. Thus, for

(25)
example, before statement (3.2.2) is used in a routine, the following statement must be executed:

\[
\text{CALL MLD (PREC(P), NAMES(X,Y,Z,A,B,C))}
\]

Only the mantissa section of each word is used in the space reserved for multi-length variables. Hence each word will contribute \(\log_{10} 2^{39} \approx 11.74\) 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 \(\equiv 512\) words). Additional store is needed for the execution of various operations; the most expensive operation is SQRT00T which requires 5 words of working store for each word of the multi-length variable in question. The multi-length routines themselves 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 \( P = \log_{10} 2^{39} \times L \approx 11.74L \) and the computing time \( T \) is given by \( T \approx N/327680 \) seconds. Thus the time required to perform a division to a precision of 750 significant decimal digits is approximately \( \frac{1}{5} \) second.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>320</td>
<td>350</td>
<td>400</td>
<td>490</td>
<td>710</td>
<td>1100</td>
<td>1800</td>
<td>3400</td>
</tr>
<tr>
<td>SUB</td>
<td>380</td>
<td>400</td>
<td>450</td>
<td>560</td>
<td>800</td>
<td>1300</td>
<td>2200</td>
<td>3900</td>
</tr>
<tr>
<td>MUL</td>
<td>320</td>
<td>400</td>
<td>650</td>
<td>1300</td>
<td>3600</td>
<td>12000</td>
<td>43000</td>
<td>160000</td>
</tr>
<tr>
<td>DIV</td>
<td>460</td>
<td>590</td>
<td>950</td>
<td>2100</td>
<td>5500</td>
<td>18000</td>
<td>62000</td>
<td>240000</td>
</tr>
<tr>
<td>POW</td>
<td>3000</td>
<td>4400</td>
<td>7800</td>
<td>18000</td>
<td>49000</td>
<td>160000</td>
<td>570000</td>
<td>2100000</td>
</tr>
<tr>
<td>IPR</td>
<td>320</td>
<td>340</td>
<td>370</td>
<td>440</td>
<td>520</td>
<td>750</td>
<td>1100</td>
<td>2000</td>
</tr>
<tr>
<td>SQRT</td>
<td>2300</td>
<td>3500</td>
<td>5000</td>
<td>8900</td>
<td>18000</td>
<td>44000</td>
<td>130000</td>
<td>430000</td>
</tr>
<tr>
<td>PRT</td>
<td>6100</td>
<td>7200</td>
<td>10000</td>
<td>18000</td>
<td>47000</td>
<td>160000</td>
<td>570000</td>
<td>2200000</td>
</tr>
<tr>
<td>REA</td>
<td>8800</td>
<td>16000</td>
<td>32000</td>
<td>80000</td>
<td>240000</td>
<td>700000</td>
<td>3000000</td>
<td>11000000</td>
</tr>
</tbody>
</table>

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 \( \log_{\frac{2 + 3}{3 + 6}} \) 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

\[ x \equiv a_i \pmod{m_i}, \quad i=1,2,\ldots,n \]  \hspace{1cm} (3.3.1)

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

Put \( M = \prod_{i=1}^{n} m_i \) and let \( y \equiv b_i \pmod{m_i} \) 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

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

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 3ULDIV (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 ([4]), we know that \( c(k) \sim \frac{\mathbb{e}^{\pi \sqrt{k}}}{k^{3/2}} \), 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 \( \theta \) be a positive real number. By its continued fraction expansion we shall mean the representation of \( \theta \) as

\[
\theta = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \ldots}}} \tag{3.4.1}
\]

where the \( a_i \) are positive integers, called the partial quotients of \( \theta \). 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 \( n > 1 \),

\[
a_n = \left[\frac{1}{r_{n-1} - a_{n-1}}\right] \tag{3.4.2}
\]

(29)
The simplest kind of continued fraction expansion occurs when \( \Theta \) is either rational or a quadratic irrational. In the former case, the sequence \( \left[ a_0; a_1, a_2, \ldots \right] \) terminates; the denominator of \( r_n \) is zero for some \( n \). In the latter case, the sequence of partial quotients is eventually periodic. When \( \Theta \) is neither rational nor a quadratic irrational, i.e. in the case when \( \Theta \) is algebraic of degree \( \geq 3 \), or transcendental, very little is known about the sequence of partial quotients. A notable exception is \( e \), whose partial quotients form the sequence \( \left[ 2; 1, 2, 1, 1, 4, 1, \ldots, 1, 2n, 1, \ldots \right] \).

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

\[
\Theta' = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \ddots + \frac{1}{a_n}}}} = \frac{p_n}{q_n}
\]

will be an approximation to \( \Theta \); the larger \( n \) is, the more accurate is the approximation \( \Theta' \) to \( \Theta \). These rational approximations are called the convergents to \( \Theta \) 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 \( n \geq 0 \)

\[
p_n = a_n p_{n-1} + p_{n-2} \quad \text{and} \quad q_n = a_n q_{n-1} + q_{n-2}
\]

The elementary properties of convergents are:

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

(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 \( \Theta \) to satisfy \( 0 < \Theta < 1 \); it is clear from the definition of the \( a_i \) that the continued fraction expansion of \( \Theta \) is essentially the same as that of \( \Theta + m \), where \( m \) is an arbitrary integer.

Hence we set
\[
\Theta = \left[0; a_1, a_2, a_3, \ldots\right]
\]
and have
\[
r_n = \left[ a_n; a_{n+1}, a_{n+2}, \ldots\right].
\]
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 \( \Theta \), \( \text{Prob}(a_i = k) = \log_2 \frac{(k+1)^2}{k(k+2)} \) \( (3.4.6) \)

2. For almost all \( \Theta \), \( \lim_{n \to \infty} \sqrt[n]{a_n} = e^{12 \log^2 2} = 3.27582 \ldots \) \( (3.4.7) \)

3. For almost all \( \Theta \), \( \lim_{n \to \infty} \sqrt[n]{a_1 a_2 \ldots a_n} = K \), 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 (13), who showed that if \( N \) existed,

\[
N > 10^{1700000}
\]  

The problem reduces to showing that the simultaneous equations

\[
\begin{align*}
3x^2 - 2 &= y^2 \\
8x^2 - 7 &= z^2
\end{align*}
\]

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

\[
(3x^2 - 2)(8x^2 - 7) = y^2 z^2 = t^2
\]

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

\[
(2 \sqrt{3})x = (1 + \sqrt{3})(2 + \sqrt{3})^m - (1 - \sqrt{3})(2 - \sqrt{3})^m
\]

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

\[
2 < m < 10^{487}
\]

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

\[ \theta = \frac{\log(2 + \sqrt{3})}{\log(3 + \sqrt{3})} = 0.74710\ 53797\ 84665\ 20012 \ldots \]

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

\[ \beta = \frac{\log \left( \frac{1 + \sqrt{3}}{8} \right)}{\log(3 + \sqrt{3})} = 0.08680\ 37805\ 12726\ 74666 \ldots \]

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

\[ \mu = 10^{487} \text{ and } K = 10^{33} \]

(4) \( \theta = \) value of \( \theta \) correct to 1040 decimal places,

\[ |\theta - \theta_0| < 10^{-1040} \]

(5) \( q \) is the denominator of the last convergent, in the continued fraction expansion of \( \theta_0 \), satisfying \( q < 10^{520} \)

(6) \( \| q \beta \| > 3 \times 10^{-33} \) and \( \| q \beta' \| > 3 \times 10^{-33} \), 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

\[ \frac{\log K M}{\log C} < m < M \]

i.e. \[ \frac{\log 10^{553}}{2 \log(2 + \sqrt{3})} < m < 10^{487} \] (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 \( \theta, \beta, \beta' \) 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
\[
\log(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots + (-1)^{n+1} \frac{x^n}{n} + \ldots \quad (-1 < x < 1) \quad (5.4.14)
\]

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

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

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

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 \( E = \text{multi-length exponential of } l \). The series (3.4.14) is then used with \( x = Y/E \).

Wrench and Shanks (12.) 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 \( q \leq 10^{520} \), namely \( q_{1002} \),

found to be

\begin{align*}
74766 & \quad 56458 & \quad 85928 & \quad 21002 & \quad 92900 & \quad 19462 & \quad 74193 & \quad 99932 \\
68435 & \quad 51834 & \quad 20544 & \quad 67033 & \quad 92527 & \quad 99010 & \quad 36030 & \quad 14382 & \quad 83128 & \quad 15409 & \quad 94079 & \quad 49641 \\
75823 & \quad 72448 & \quad 20294 & \quad 43561 & \quad 15091 & \quad 97552 & \quad 65496 & \quad 09837 & \quad 65725 & \quad 70805 & \quad 71781 & \quad 3765 \\
90201 & \quad 82968 & \quad 04828 & \quad 89690 & \quad 91216 & \quad 09036 & \quad 42656 & \quad 74598 & \quad 43126 & \quad 05161 & \quad 50601 & \quad 13889 \\
48311 & \quad 34448 & \quad 43630 & \quad 77762 & \quad 01998 & \quad 69513 & \quad 73685 & \quad 70540 & \quad 20065 & \quad 06420 & \quad 17453 & \quad 43949 \\
32542 & \quad 08937 & \quad 00733 & \quad 92823 & \quad 67336 & \quad 28270 & \quad 20008 & \quad 54767 & \quad 81468 & \quad 64873 & \quad 46464 & \quad 28193 \\
39455 & \quad 78382 & \quad 27505 & \quad 86507 & \quad 22688 & \quad 57730 & \quad 19978 & \quad 42556 & \quad 32569 & \quad 44952 & \quad 91835 & \quad 82629 \\
52538 & \quad 66886 & \quad 97685 & \quad 22768 & \quad 40339 & \quad 96403 & \quad 83429 & \quad 92464 & \quad 53396 & \quad 46774 & \quad 48258 & \quad 60409 \\
41197 & \quad 29139 & \quad 39485 & \quad 18564 & \quad 04207 & \quad 26381 & \quad 80339 & \quad 63053 & \quad 74225 & \quad 67257 & \quad 33135 & \quad 04814 \\
\end{align*}
We remark that, referring to the asymptotic relation (3.4.7), the solution for $x$ of the equation
\[
\frac{n^2 x}{e^{12 \log 2}} = 10^{520}
\]
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 $\beta$ and $\beta'$ respectively, and taking the fractional part, we obtain
\[
\|q \beta\| = 0.42279 62795 75983 04235 \ldots
\]
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
\[
x^3 - 6x^2 + 4x - 2 = 0 \quad (3.5.1)
\]
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

\[
x^3 - ax - b = 0 \quad a, b \text{ integral and } |a|, |b| \leq 9.
\]

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 \( 3.31862 \ 82177 \ldots \) 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 \( \Theta \), the probability that in the continued fraction expansion of \( \Theta \) a particular partial quotient \( a_i \) equals \( k \) is

\[
\log_2 \left( \frac{(k+1)^2}{k(k+2)} \right).
\]

Hence, for almost all \( \Theta \),

\[
\text{Prob}(a_i \geq k) = \sum_{m=k}^{\infty} \log_2 \left( \frac{(m+1)^2}{m(m+2)} \right) = \log_2 \left( \frac{1+1}{k} \right) \approx \frac{1.44}{k}.
\]

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
\[
f(\tau) = q^{\frac{1}{24}} \prod_{n=1}^{\infty} \left(1 + q^{2n-1}\right)
\]
where \( q = e^{2\pi i \tau} \).

To illustrate the theory of class-invariants constructed by Weber (see Weber(10)), let \(-n\) be a negative integer with \( n \equiv 3 \pmod{8} \). Let \( h \) be the class-number of the determinant \(-n\) and let
\[
ax^2 + by^2 + cz^2 \quad (k = 1, 2, \ldots, h)
\]
be a complete set of reduced imprimitive binary quadratic forms with determinant \( b_k^2 - 4akc_k = -n \). To construct such a set, one takes
\[
b = \pm 1, \pm 3, \pm 5, \ldots \text{ with } |b| \leq \sqrt{\frac{1}{3}n}
\]
and then expresses \( \frac{1}{2}(b^2 + n) \) as a product of positive factors \( a \) and \( c \) in all possible ways with either \( c > a \) and \(-a < b < a\) or \( c = a \) and \( 0 \leq b \leq a \). Clearly the form
\[
x^2 + xy + \frac{1}{2}(n+1)
\]
will belong to this set.

Let now \( \tau_k \) be that root, with positive imaginary part, that satisfies the equation
\[
a_kx^2 + b_kx + c_k = 0.
\]
Then, by Weber (10), 418-423, the equation
\[
\prod_{k=1}^{h} (x - j(\tau_k)) = 0
\]
has integral coefficients.

Furthermore, we have
\[
j\left(\frac{-1 + \tau}{2}\right) = -\left(\frac{f^{24}(\tau) - 256}{f^{24}(\tau)}\right)^3.
\]
From (3.5.2), one of the \( \tau_k \) must be \( \frac{-1 + \sqrt{-n}}{2} \). Hence \( f^{24}(\sqrt{-n}) \) must satisfy

(37)
an algebraic equation of degree $3h$ since, from (3.5.3), $j(-\frac{1 + \sqrt{-2347}}{2})$ satisfies an algebraic equation of degree $h$. It can be shown that this equation of degree $3h$ reduces to an equation

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

with integral coefficients where the 24th powers of the roots $\alpha_k$ 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 + 12542000088928251612025910348856556695501877957727675109376000y^4 + 3850392729356723089776031377897241173629209744575099844793532416000000y^3 - 397498016421209870835347342908155232090445272713278332932508876800000000y^2 - 2750056058932258722835289220636084306945561562768851721342439915520000000000y + 9016445699225621599080750526731943915918135938859768906808033280000000000000 = 0,$

with discriminant


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

(38)
Similarly by \((3.5.4)\), \(x = f(\sqrt{-2347})\) will satisfy an equation of degree 15, namely
\[
x^{15} - 566x^{14} - 950x^{13} - 2x^{12} + 1676x^{11} + 1688x^{10} - 1216x^9 \\
- 5080x^8 - 3520x^7 + 1136x^6 + 3056x^5 + 5888x^4 - 3728x^3 \\
- 9152x^2 - 224x - 32 = 0 .
\]
The discriminant of this equation is
\[-2^{102}.3^2.5^6.7^4.13^{10}.43^2.73^2.109^2.409^2.2347^7\]
and the real root is
\[
567.67349 \; 42228 \; 67666 \ldots = e^{\frac{i\pi}{12}\sqrt{2347}} \prod_{n=1}^{\infty} \left(1 + e^{-(2n-1)i\pi\sqrt{2347}}\right).
\]
We now fix \(D = -d = -163\). Here the class-number \(h(-163) = 1\), with \(y = j(-1 + \frac{1}{2})\) satisfying the linear equation
\[
y + (2^6.3.5.23.29)^3 = 0 \quad (3.5.5)
\]
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 \(j(R)^3\) 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 \(\Theta\), is
\[
e^{\frac{i\pi}{12}\sqrt{163}} \prod_{n=1}^{\infty} (1 + e^{\frac{1}{3n\pi\sqrt{163}}}) \ldots \quad (3.5.6)
\]
The main factor of this expression is \(e^{\frac{i\pi}{12}\sqrt{163}}\), which approximates \(\Theta\) with an error of less than \(10^{-17}\). The continued fraction expansion of \(e^{\frac{i\pi}{12}\sqrt{163}}\) 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 \((3.5.5)\)
\[ j\left(-\frac{1 + \sqrt{-163}}{2}\right) = -(2^6 \cdot 3 \cdot 5 \cdot 23 \cdot 29)^3 = N, \text{ say.} \]

From the definition of \( j \),
\[
j(\tau) = \sum_{n=-1}^{\infty} c(n)x^n = x^{-1} + 744 + 196884x + 21493760x^2 + \ldots,
\]
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
\[ N = -e^{\pi\sqrt{163}} + 744 - 196884e^{-\pi\sqrt{163}} + 21493760e^{-2\pi\sqrt{163}} - \ldots \]

Thus
\[ e^{\pi\sqrt{163}} = -N + 744 - 196884e^{-\pi\sqrt{163}} + 21493760e^{-2\pi\sqrt{163}} - \ldots \quad (3.5.7) \]

Since, by Lehmer (2),
\[ c(k) \sim \frac{e^{4\pi\sqrt{k}}}{2^k \cdot k^2} \quad \text{as} \quad k \to \infty \]
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
\[ X = e^{\pi\sqrt{163}} = 262,537,412,640,768,743,999,999,999,999,9250 \ldots \]

We now show that \( X^2 \) is nearly an integer; which is not trivially obvious. Let \( M = -N + 744 \). From (3.5.7)
\[ X = M - 196884X^{-1} + 21493760X^{-2} - \ldots \]

Hence, multiplying both sides by \( X \),
\[ X^2 = MX - 196884 + 21493760X^{-1} - \ldots \]
\[ = M(M-196884X^{-1} + 21493760X^{-2} - \ldots) - 196884 + 21493760X^{-1} - \ldots \]
\[ = M^2 + M(-196884X^{-1} + 21493760X^{-2} - \ldots) - 196884 + 21493760X^{-1} - \ldots \]

(40)
\[
\begin{align*}
&= x^2 + (x+196884x^{-1} - 21493760x^{-2} + \ldots)(-196884x^{-1} + 21493760x^{-2} - \ldots) \\
&\quad - 196884 + 21493760x^{-1} - \ldots \\
&= x^2 - 393768 + 42987520x^{-1} + o(x^{-2})
\end{align*}
\]
We therefore deduce that \( e^{2\sqrt{163}} \) should differ from an integer by approximately
\[
\frac{42987520}{262537412640768744} \approx 1.6373 \times 10^{-10},
\]
the error being one of excess.

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

<table>
<thead>
<tr>
<th>Power of ( X )</th>
<th>Fractional part</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X )</td>
<td>0.99999 99999 99250 ...</td>
</tr>
<tr>
<td>( X^2 )</td>
<td>0.00000 00001 63738 ...</td>
</tr>
<tr>
<td>( X^3 )</td>
<td>0.99999 99901 23693 ...</td>
</tr>
<tr>
<td>( X^4 )</td>
<td>0.00000 03084 64322 ...</td>
</tr>
<tr>
<td>( X^5 )</td>
<td>0.99999 36541 87468 ...</td>
</tr>
<tr>
<td>( X^6 )</td>
<td>0.00009 71752 54162 ...</td>
</tr>
<tr>
<td>( X^7 )</td>
<td>0.99880 93165 26134 ...</td>
</tr>
<tr>
<td>( X^8 )</td>
<td>0.01223 41690 69154 ...</td>
</tr>
<tr>
<td>( X^9 )</td>
<td>0.29886 26339 54035 ...</td>
</tr>
</tbody>
</table>

The above analysis shows that, in particular, the first few odd
powers of \( X = e^{\sqrt{163}} \) are nearly integers. Now, as observed previously, the
continued fraction expansion of \( e^{\sqrt{163}} \) is unremarkable and indeed it is
found that the values of the partial quotients first differ from the partial
quotients of \( \Theta \) at the 16th. term, immediately before the first large term
occurs. This implies that the first factor ignored, namely
\[
1 + e^{-\sqrt{163}}
\]
(3.5.8)
is, in some sense, responsible for the first large term in the continued

\[(41)\]
fraction expansion of $\Theta$. Now (3.5.8) may be written as

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

where

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

and

$$N_1 = 133 \, 34624 \, 07511$$

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

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

where

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

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

$$M_3 \simeq 1.24 \times 10^{67}, \quad N_3 \simeq 1.56 \times 10^5$$

$$M_4 \simeq 8.6 \times 10^{121}, \quad N_4 \simeq 800.$$

To test the hypothesis that the very large partial quotients in the continued fraction expansion of $\Theta$ 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

$$\Theta_1 = x^{1/2}(1 + z_1^{-1})(1 + z_3^{-1})(1 + z_5^{-1}) \cdots$$

(3.5.9)

where $z_n = [x^n + 1/2]$, so that $z_n$ is the nearest integer to $x^n$. The value of $\Theta_1$ so obtained is extremely close to $\Theta$, the error being approximately $5.8 \times 10^{-47}$. Despite this minute numerical change, the effect on the continued fraction expansion of $\Theta$ is so drastic that only the first large term remains and the remaining large partial quotients disappear. When the

\[ (42) \]
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 \( f(\sqrt[3]{-163}) \);
2. the field \( \mathbb{R}(\sqrt[3]{-163}) \) obtained by adjoining \( \sqrt[3]{-163} \) to the rationals has class-number 1;
3. the root \( \theta \) of the equation is approximated to seventeen places of decimals by \( \theta' = e^{\frac{2\pi i}{3\sqrt[3]{163}}} \);
4. the ratio \( \theta'/\theta \) is given by the infinite product
\[
\prod_{n=1}^{\infty} \left(1 + e^{-(2n-1)\pi i \sqrt[3]{163}}\right);
\]
5. the first few terms of this product may all be written in the form
\[
1 + \frac{1}{M_1 + 1} + \frac{1}{N_1 + 1} + \cdots
\]
where \( M_1 \) is "very large" and \( N_1 \) is "large";
6. the presence of such factors in (5) produces large partial quotients in the continued fraction expansion of \( \theta \). 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 \( \theta \) is extremely small.

On the basis of these observations we can make a prediction. We have seen that \( e^{\frac{2\pi i}{3\sqrt[3]{163}}} \) 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 $\Theta$ to persist beyond about the 170th. term, the point where the factor $1 + e^{-9\mu_{183}}$ could be expected to have an effect. This prediction is borne out. A computation of the first 875 partial quotients of $\Theta$ 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 $\Theta$ do not themselves exhibit any remarkable arithmetical properties. An arithmetical property can, however, be obtained in the following manner.

The convergents of $\Theta$ form an infinite sequence

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

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

$$\frac{-7}{1}, \frac{10}{3}, \frac{-119}{22}, \frac{1002}{91}, \frac{-163}{204}, \ldots$$

in which every other member has the same sign (which follows from the theory of convergents) and each member is of the form $\frac{m}{n^3}$, $m$ and $n$ integral and $m > 0$, since $\Theta$ 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 $\Theta$. 

(44)
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 $\Theta$. One can expect this, since, at this position, we have a convergent

\[ (45) \]
which, in relation to the size of its denominator, is a good approximation
to $\Theta$.

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

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

The corresponding values of $m_i$ begin

\[
\begin{align*}
m_{16} &= 1 85801 \\
m_{32} &= 206 32262 06688 99807 \\
m_{58} &= 34354 87084 12692 17891 33839 29164 09801
\end{align*}
\]

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

\[
\begin{align*}
m_{80} &\approx 1.28 \times 10^{49} \\
m_{102} &\approx 6.88 \times 10^{63} \\
m_{120} &\approx 7.72 \times 10^{74} \\
m_{138} &\approx 2.97 \times 10^{93} \\
m_{160} &\approx 1.02 \times 10^{107}
\end{align*}
\]

The attempted factorisation of these numbers reveals that

\[
\begin{align*}
m_{16} &= 7 \cdot 11 \cdot 19 \cdot 127 \\
m_{32} &= 11 \cdot 19 \cdot 127.\ast \\
m_{58} &= 7 \cdot 19 \cdot 127.\ast \\
m_{80} &= 5 \cdot 7 \cdot 11 \cdot 127.\ast \\
m_{102} &= 7^2 \cdot 11 \cdot 127.\ast \\
m_{120} &= 7^2 \cdot 11 \cdot 127 \cdot 419 \cdot 1093.\ast \\
m_{138} &= 7 \cdot 11 \cdot 19 \cdot 127.\ast \\
m_{160} &= 7 \cdot 11^2 \cdot 19 \cdot 127.\ast
\end{align*}
\]

(46)
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 $\sqrt[3]{j(\tau)-1728}$ when $\tau = \frac{-1 + \sqrt{-163}}{2}$. For we have, by (3.5.5)

$$j(\tau) = \left(2^6 \cdot 3 \cdot 5 \cdot 23 \cdot 29\right)^3$$
$$= -262537412640768000$$

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

Hence
$$\sqrt[3]{j(\tau)-1728} = 40133016\sqrt[3]{-163}$$
$$= 2^3 \cdot 3^3 \cdot 7 \cdot 11 \cdot 19 \cdot 127\sqrt[3]{-163}.$$  

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

$$e^{\pi \sqrt{67}} = 14.7194952743.999998662454224\ldots$$
and $$e^{\pi \sqrt{43}} = 884736743.999777466034906\ldots$$

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

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

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 $\Theta$ and $\Theta_1$ to 200 decimal places, together with a table of the partial quotients $a_j$ and $a'_j$ in their respective continued fraction expansions.
\[
\theta = 5.31862 82177 50185 65910 96801 53318 02246 77219 19808 83690
\]
\[
0.2602 28091 99584 01958 97457 32187 43665 34591 07487 15400
\]
\[
45589 07647 42444 78645 91488 72327 64878 31165 98454 79445
\]
\[
12414 29908 75700 21982 39534 04098 41477 60189 42443 29911
\]
\[
\theta_1 = 5.31862 82177 50185 65910 96801 53318 02246 77219 19808 77903
\]
\[
25291 02486 16405 49342 75592 94236 99171 57852 32639 76230
\]
\[
30208 06883 94087 98203 79559 02991 44546 73822 47205 03958
\]
\[
81415 44232 97654 81169 19582 67166 38540 80329 70950 41676
\]

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

(48)
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 $k \geq 4$ by

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

(49)
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

\[
A_1 = \{ z : |z| \geq 1, -\frac{1}{2} \leq Rz \leq 0 \}, \\
A_2 = \{ z : |z| > 1, 0 \leq Rz < \frac{1}{2} \}, \\
A_3 = \{ z : z = \infty \}.
\]

It can be shown that any member of $M_k$ has $\frac{k}{12}$ zeros in $F$, if zeros at $z=1$ 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 $4 \leq k \leq 26$, 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 $k \geq 4$, 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 $y = \frac{1}{2} \geq 1$.

(i.e. the conjecture holds for this case)
(2) if \( k \equiv 0 (\text{mod} \ 4) \) and \( y = \frac{4}{3} z \)
then \( S_k(z) \neq 0 \) providing that \( y > 1 + \frac{1}{2\pi} \log C_k \)
where \( C_k = \frac{\alpha_k}{\alpha_k^{2}} \), and \( \alpha_k = \frac{(2\pi)^{k} \zeta(k)}{(k-1)!} \)

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

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

Define \( a_n(\nu) \) by

\[
j(\nu) = x^{-\nu} \sum_{n=0}^{\infty} a_n(\nu) x^n; \ \nu \geq 0
\]

and let

\[
s_\nu = a_\nu(\nu) - 24 \sum_{m=1}^{\nu} a_\nu(\nu) \sigma(m)
\]

where

\[
\sigma(m) = \sum_{d|m} d
\]

Finally, for \( n \geq 0 \), let

\[
\Delta_n = \begin{vmatrix}
\varepsilon_0 & \varepsilon_1 & \varepsilon_2 & \ldots & \varepsilon_n \\
\varepsilon_1 & \varepsilon_2 & \varepsilon_3 & \ldots & \varepsilon_{n+1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\varepsilon_n & \varepsilon_{n+1} & \varepsilon_{n+2} & \ldots & \varepsilon_{2n}
\end{vmatrix}
\]

(51)
We find that:

\[ \Delta_0 = 1 \]

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

\[ \Delta_2 = \begin{vmatrix} 1 & 720 & 911520 \\ 720 & 911520 & 1301011200 \\ 911520 & 1301011200 & 1958042030400 \end{vmatrix} = 27454623356160000 . \]

By comparing the determinants \( \Delta_n \) with a sum of products

\[ \prod_{1 \leq j < k \leq n} (\alpha_i - \alpha_j)^2 \]

where \( \alpha_0, \alpha_1, \ldots, \alpha_n \) assume all sets of \( n+1 \) values \( j(\xi) \) taken from the zeros 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 \( k > k_0 \), \( E_k \) 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 \( \Delta_n \), \( n=0,1,2,\ldots,13 \), 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 \( \Delta_n \). A table of the factorisation of \( \Delta_1, \Delta_2, \ldots, \Delta_{13} \) is given below. It lists for each prime \( p \) dividing \( \Delta_n \) the power of \( p \) in the factorisation; a blank entry denotes zero.
One fact immediately noticeable from the above table is that, for \( n=1,2,\ldots,13 \), \( \Delta_{n-1} \) divides \( \Delta_n \). Let \( \Delta'_n = \frac{\Delta_n}{\Delta_{n-1}} \), \( n \geq 1 \).

Thus 
\[ \Delta'_1 = 393120 \]
\[ \Delta'_2 = 69837768000 \, . \]

The factorisation of \( \Delta'_n \), \( n=1,2,\ldots,13 \) is given in the first 13 columns of the table below.
To obtain more data, the Chinese Remainder Theorem was used to compute
\[ \Delta'_1, \ldots, \Delta'_{25}, \]
instead of computing the \( \Delta'_n \) 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 \( \Delta_n \) was then obtained with its elements \( g_0, \ldots, g_{2n} \) computed mod \( p \). To obtain a list of the \( \Delta'_n \), it was necessary to set up \( \Delta'_N \), where \( N \) was the largest \( n \) for which \( \Delta'_n \) was to be computed — in this case \( N \) was 25. Reducing \( \Delta'_N \) to a triangular matrix by Gauss elimination then gave the \( \Delta'_n \) 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 \( \Delta'_n \mod \) the product of these primes. Just how many moduli need to be used in an application of

\( (55) \)
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 $\Delta'_n$ 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 $\Delta_n$ was divisible by $\Delta_{n-1}$ for $n \geq 1$. The results, as it turned out, clearly suggested that the $\Delta'_n$ were positive integers and the factorisation of these numbers for $1 \leq n \leq 25$ 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 $\Delta''_n = \Delta'_n / \Delta'_{n-1}$, $n > 2$. Thus $\Delta''_2 = 2 \cdot 5^2 \cdot 11 \cdot 17 \cdot 19$, $\Delta''_3 = 2 \cdot 3 \cdot 5^2 \cdot 23 \cdot 29 \cdot 31 \cdot 37$, and $\Delta''_4 = 3 \cdot 5 \cdot 7 \cdot 41 \cdot 43$. From a table of $\Delta''_2, \ldots, \Delta''_{25}$ the author conjectured that, for $n \geq 2$,

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

If this conjecture is true, then obviously $\Delta''_n > 0$ for $n \geq 0$ 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
\[ j(z) = x^{-1} + 744 + 196884x + \ldots. \]

If $\sqrt{E_4}(z)$ is expanded as a power series in $j^{-1}$, i.e. if
\[ \sqrt{E_4} = 1 + \sum_{n=1}^{\infty} a_n j^{-n} \]
then the $a_n$ enjoy a property similar to that found for the $\Delta_n$ in (3.6.1).

From the fact that $\sqrt{E_4}$ is a hypergeometric function of $12^{3/4}/j$, it can be shown that if, for $n > 1$, $b_n = \frac{a_n}{a_{n-1}}$ with $a_0 = 1$, then
\[ b_n = \frac{12(12n - 7)(12n - 11)}{n^2} \quad (3.6.3) \]

(2) By a change of variable, it may also be shown that if
\[ \sqrt{E_4} (1 - 1728j^{-1})^{-1/2} = 1 + \sum_{n=1}^{\infty} a_n' j^{-n} \]
and $b'_n = \frac{a'_n}{a'_{n-1}}$, then
\[ b'_n = \frac{12(12n - 1)(12n - 5)}{n^2} \quad (3.6.4) \]

(3) Finally, if
\[ \sqrt{E_4} = 1 + \sum_{n=1}^{\infty} a''_n j^{-n} \]
and $b''_n = \frac{a''_n}{a''_{n-1}}$, then
\[ b''_n = \frac{24(6n - 1)(6n - 5)(2n - 1)}{n^3} \quad (3.6.5) \]

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(3)).

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.

\[
\text{IF (I) 1,}
\]

is equivalent to

\[
\text{IF (I) 1,0,0}
\]

where a 0 label indicates the next statement. Similarly

\[
\text{IF (I),1, is equivalent to IF (I) 0,1,0}
\]
(2) Logical IF statement.

\[
\text{IF (L) 1,}
\]

\text{next statement}

Where \( L \) is a logical expression, is equivalent to

\[
\text{IF (L) GO TO 1}
\]

and \( \text{IF (L),1} \) is equivalent to

\[
\text{IF (L) GO TO 2}
\]

\[
\text{GO TO 1}
\]

\text{2 next statement}

(3) DO - loop.

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

\[
\text{DO 1 I = N1,N2,N3}
\]

\[
\cdot
\]

\[
\cdot
\]

\text{1 statement}

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

(4) FOR - loop

\[
\text{FOR I = N1,N2,N3}
\]

\[
\cdot
\]

\[
\cdot
\]

\text{REPEAT}

is identical to

\[
\text{DO 1 I = N1,N2,N3}
\]

\[
\cdot
\]

\[
\cdot
\]

\text{1 statement}
except that if $N_2 < N_1$, the for-loop will not be executed and the do-loop will be executed once (assuming throughout that $N_3$ is positive).

(5) Array storage.

Arrays or variables coded next to each other are stored contiguously. 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 \( \pi \) comment
```

is equivalent to

```
statement
```

```
C comment
```

(62)
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

$$I_1^{(I_2)} \cdot I_2^{(I_3)} \cdot I_3^{(I_4)} \cdot \cdots \cdot I_{I_2-1}^{(I_{I_2})},$$

followed by an asterisk if I4 is non-zero. (Here $I_k^{(I)} = I(k)$).

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.

**INFSEL (called from MLJ)**

CALL INFSEL(N) where N is an integer constant or variable will select input 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

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

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).


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.


(66)


24. X. See Siegel (17).
4.5. THE SUBROUTINE LIBRARY.

ABBREVIATIONS

THE RESTRICTION 1 ≤ N ≤ 2**35 - 1 is denoted by R*N

THE RESTRICTION 0 ≤ N ≤ 2**35 - 1 is denoted by R0(N)

THE RESTRICTION 0 ≤ I, J ≤ 2**35 - 1 is denoted by RMOD(M, I, J)
1 ≤ M ≤ 2**35 - 1
(I*J)/M < 2**39.

THE RESTRICTION 0 ≤ I ≤ 2**35 - 1 is denoted by RMOD(M, I)
1 ≤ M ≤ 2**35 - 1

THE RESTRICTION M√(2/3) < F is denoted by REUL(M, N)

THE RESTRICTION M√(2/N1 (3N11)) < F is denoted by REULQ(M, N, NN)

THE RESTRICTION 'MUST NOT BE USED IN A PRINT STMT.' is denoted by RPRINT

THE RESTRICTION 'MUST NOT BE USED IN AN IF STMT.' is denoted by RIF

F := 2**36 + 1

F(x) = \prod_{m=1}^{∞} (1 - x**m)

ML = MULTI-LENGTH = VARIABLE PRECISION.
SUBROUTINE BQP(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), I : 1,N IS CALCULATED FOR DISCRIMINANT -V. ON RETURN FD MAY ASTERISK
C Z = CLASS NUMBER OF DISCRIMINANT -V. IF IPRINT NEQ 0, FORMS ARE 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 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 X(7),Q(7),E0,E(7),F(200),FF(200)
TEXT STAR,STARS(500)
INTEGER FA(I),FB(I),FC(I),FD(I)
FORMAT(5H0D") 101 FORMAT(I6,6X,4HH = ,14)
102 FORMAT(5,1X,316,A5))
IE(III) 3,,3
RE I N D 2:
READ. TAPE 2,F1
D0 4,P =1,200
4 FB(P) = F(P)*F(P)
IE(IUNLD) 3,
CALL UNLOAD(2)
3 D = V
IE(D) 2,2,
LE = IZR(F4,D) 3)
IE(LL) 4,
PRINT 8:
FORMAT(8H BQP: 5x,14H=D NEQ 3 MOD 4)
1 RETURN
4 D4 = 4*D
RJX:N = 0
P = 2:
5 LE = IZR(F(P),D)
IE(LL) 7*,7
LE = IZR(FF(P),D)
IE(LL) 7*,7
R = R * 1
Q(R) = F(P)
P = P + 1
IF (3*FF(P) = -D) 5,5,
IF (R) 43,43
JX = 0
GO TO 44;
JX = 1: 
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
LB =: IZR(F(P),H)
IE(LL) =: 14,
P =: P + 1
IE(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 =: IZR(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
IE(JA) ,35.
FOR JH =: 1,JA.
A =: A*X(E(1))
REPEAT'
IE(JB) ,36.
FOR JH =: 1,JB.
A =: A*X(E(2))
REPEAT'
IE(JC) ,37.
FOR JH =: 1,JC.
A =: A*X(E(3))
REPEAT'
IE(JD) ,36.
FOR JH =: 1,JD.
A =: A*X(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 LE := IZR(Q(G), B)
    IF(LL) 26, 26:
    LE := IZR(Q(G), A)
    IF(LL) 26, 29, 28:
28 G := G + 1
    IF(Q(G) = B) : 26, 26, 30
29 LB := IZR(Q(G), C)
    IF(LL) 26, 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
35 REPEAT
36 REPEAT
37 REPEAT
38 REPEAT
39 REPEAT
40 REPEAT
41 REPEAT
42 REPEAT
43 B = B + 2.
44 IF (3*B-B1 = D) 10, 10,
45 IF (IPRINT), 2;
46 PRINT 101, -D, Z
47 NN = (N+4)/5:
48 DO 51 MM = 1, NN
49 PRINT 102, (FA(M), FB(M), FC(M), STARS(M), M = MM, N, NN)
50 RETURN
51 END
SUBROUTINE COEFF(X,M,Y,N,K,Z,MM)

PURPOSE: Z: COEFFICIENT OF A**K (MOD MM) IN X*Y WHERE:

X := A**M \ldots \quad Y := A**N \ldots

ARGUMENTS, X, Y INTG ARRAYS. MM IS MODULUS:

RESTRICTIONS: RMOD(MM,X(i),Y(K-I)), I = M..K..N.

(K+1..N-M)*MM < F

CALLS IZMR, IZR

B-REGS, 0..11

ELEMENTS OF X ACCESSED: M TO K..N

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, 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 CONTAINING
C NUM/DEN = XX. PRINTS CONTINUED FRACTION IF NTEST NEQ 0

B-REGS, 0 := 10
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, 112, F17.10)
102 RETURN
END
SUBROUTINE DCMODM(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) ... IX(2*NN-1) = 0 (MOD MM). OTHERWISE: ID NEG 0 (FAULT GIVES: ID = -1).

C RESTRICTIONS: NN < 500, R*(MM)

C CALLS IJM0DK, IZR

C B-REGS, O = 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)) = 0,8
    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 = 1,N-1
    Y(I) = IJM0DK(M,0,I+1,X(I+1))
    LX = N:
    LY = N+1:
    DO 5: K = 1,N-10
        LXY = LX-LY:
        IF(LXY) 6,14
        F0R I = 0,LXY=1:
            X(I) = IJM0DK(M,0,Y(LY),X(I))
            REPEAT
            DO 3: I = LXY,LX-1:
                IA = IJM0DK(M,0,X(I),Y(LY)) - IJM0DK(M,0,X(LX),Y(I-LXY))+M
                X(I) = IZR(M,IA)
                LX = LX-1:
                IF(LX) 11,11
                IF(X(LX)) 6,6
                LMX,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:


11 CONTINUE!

10 ID = X(0)
RETURN
6 LXY = LY-1,LX
   IF(LXY) 24,6
   FOR I = 0,LXY=1:
       Y(I) = IJM0DK(M,0,X(LX),Y(I))
       REPEAT
DO 4 I = LXY, LY = 1
IA = IJMUDK(M, 0, Y(I), X(LX)) = IJMUDK(M, 0, Y(LX), X(I-LXY)) + M
Y(I) = IZR(M, IA)
LY = LY = 1:
IF (LY) 12, 12, 1
IF (Y(LY)) .5, .5
LX, LY = LY = 1
IF (LY) 12, 12, 1
DO 10 I = LLX, 0, -1
IF (Y(I)) .5, .5:
LY = LY = 1:
IF (LY) 12, 12, 1
10 CONTINUE
12 ID = Y(0)
RETURN
5 CONTINUE
5 ID = -1
PRINT 13:
FORMAT (8H DOOMDM..5X, 36HFPUIT.. DISCRIMINANT SET TO -1)
RETURN
END
SUBROUTINE DETMOD(A, IA, NN, IDET, MM)

PURPOSE. COMPUTES DETERMINANT OF MATRIX A

ARGUMENTS. A. IS NN BY NN INTEGER MATRIX; IA. IS FIRST DIMENSION OF A IN

CALLING PROGRAMME; IDET SET TO DETERMINANT; COMPUTED MOD MM.

RESTRICTIONS. R*(MM)

CALLS IJMODK, IZR, RECIPR

B-REGS. 0 = 11

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 1 = N + 1, NN

IF(A(I, N)), 6, 6

5 CONTINUE!

IDET = 0

PRINT 12:

12 FORMAT(8H DETMOD, 8X, 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

LD = IJMODK(M, 0, LL, 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, LL, A(N, N))

IDET = L

RETURN

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

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

        INTEGER X(1), Y(1), A(1), W
        I = IZI(2, MM)
        IY = IZI(Y(N), MM)
        NA = M = N
        DO 1: I = M, NUM*M+1
        X(I) = IZMR(M, X(I), I+MM)
        X(I) = IZMR(M, X(I), MM-IY)
        DO 2: I = 1, NUM
        W = 0
        FOR J = 2: I
          W = W + IZMR(M, Y(N+J-1), A(I-J+1))
          IZR(M, W)
        A(I) = IZMR(M, IY, X(M+I-1)-W+MM)
        RETURN
        END
SUBROUTINE DISCUB(A1, A2, B1, B2, C1, C2, D, MM)

PURPOSE: COMPUTES DISCRIMINANT OF CUBIC


NEGATIVE IF NECESSARY. D IS SET TO DISCRIMINANT. MM IS MODULUS.

RESTRICTIONS: M MUST BE RELATIVELY PRIME TO A2, B2, C2.

CALLS IJM0DK, IZI, IZR

B-REGS: 0 - 9:

IF A = A1/A2, B = B1/B2, C = C1/C2, THEN

\[ D = A, B = 4B - 27C + 18ABC - 4AC \]

INTEGER A1, A2, B1, B2, C1, C2
INTEGER A, B, C, AA, AAA, BB, BBB, CC, D
MM = MM
A = IJM0DK(M, 0, A, IZI(A2*M, M))
B = IJM0DK(M, 0, B, IZI(B2*M, M))
C = IJM0DK(M, 0, C, IZI(C2*M, M))
AA = IJM0DK(M, 0, A, AA)
AAA = IJM0DK(M, 0, A, AA)
BB = IJM0DK(M, 0, B, BB)
BBB = IJM0DK(M, 0, B, BB)
CQ = IJM0DK(M, 0, C, C)
I = IJM0DK(M, 0, AA, BB) - IJM0DK(M, 0, 4, BBB) - IJM0DK(M, 0, 27, CC)
AA = IJM0DK(M, 0, 18, A)
AA = IJM0DK(M, 0, AA, B)
AA = IJM0DK(M, 0, AA, C)
CQ = IJM0DK(M, 0, 4, AAA)
CQ = IJM0DK(M, 0, CC, C)
I = I * AA * CC * M + M
D = IZR(M, I)
RETURN
END
SUBROUTINE EULD(I, J, N, M, MM)

PURPOSE: CALLS EULDIV, EULDIE, QEULDI OR QEULDE, depending on arguments.

ARGUMENTS: I, J INTEGER ARRAYS.

CALL EULD(I, J, N) WILL GIVE J = I / F(X) TO N TERMS, EXACTLY.
CALL EULD(I, J, N, M) WILL CALL EULDIV WITH THESE ARGUMENTS IF M > 0.
CALL EULDIE IF M = 0.
CALL: EULD(I, J, N, M, MM) WILL CALL QEULDI WITH THESE ARGUMENTS IF M > 0.
QEULDE IF M = 0.

I(0), J(0) OVERWRITTEN.

RESTRICTIONS: REUL(M, N), REULQ(M, N, MM), RO(M)

CALLS EULDIV, EULDIE, QEULDI, QEULDE

B-REGS, 1: 60 = 64

SUBROUTINE EULM(I, J, N, M, MM), MUTATIS MUTANDIS, CALLING:

EULMUL, EULMUE, QEULMU, QEULME

WRITTEN IN ASP
SUBROUTINE EULDIV(I, J, N, M)

C PURPOSE: DIVIDES \( I = 1 + I(1) x + \cdots \) BY \( f(x) \) GIVING \( J = 1 + J(1) x + \cdots \)

C ARGUMENTS: M IS MODULUS. N = NUMBER OF TERMS. I(0), J(0) OVERWRITTEN.

C RESTRICTIONS: R(M), REUL(M, N).

FORTRAN VERSION OF AN ASP ROUTINE.

DIMENSION I(1), J(1)
FOR L = 1, N
  JJ = I(L)
  K3 = L-1
  K2 = L-2
  K4 = -1
  IF (K2) 2, 2, 3
  K4 = K4+12
  JJ = JJ + J(K1) + J(K2)
  K3 = K3+12
  K2 = K2+K4
  K4 = K1+K3
  G0 TO 4
  IF (K2) 12, 12, 13
  K4 = K4+12
  JJ = JJ + J(K1) + J(K2)
  K3 = K3+12
  K2 = K2+K4
  K4 = K1+K3
  G0 TO 14
  IF (K2) 11, 32, 32
  JJ = JJ + J(K1) + 1
  G0 TO 18
  IF (K1) 19, 16, 17
  JJ = JJ + J(K1)
  G0 TO 19
  JJ = JJ + J(K1)
  JJ = JJ + J(K1)
  IF (I(L)) 19, 20, 20
  JJ(L) = JJ + J(L) + M
  REPEAT
SUBROUTINE EULMUL(I, J, N, M)

0. PURPOSE: MULTIPLIES \( I = 1 \times I(1) \times \ldots \) BY \( F(X) \) GIVING

0. \( J = 1 \times J(1) \times \ldots \)

0. ARGUMENTS: \( M \) IS MODULUS, \( N \) = NUMBER OF TERMS, \( I(0), J(0) \) OVERWRITTEN;

0. RESTRICTIONS: \( R^{+}(M), R^{+}(M, N) \).

0. TIME ~ \( N^{3/2} \)

0. FORTRAN VERSION OF AN ASP ROUTINE:

DIMENSION I(1), J(1)
M6 = 6
M5 = 5
DO 20 L = 1, N
J0 = I(L)
K = 1
K1 = L - 1
K2 = K1 + K1
IF(K2) 2, 2,
J0 = JJ - I(K1) - I(K2)
K1 = K1 + K1
M6 = M5
K = K + 2
G0 TO 4
2 IF(K2) 1,
J0 = JJ - I(K1) - 1
G0 TO 8
1 IF(K1) 8, 7
J0 = JJ - 1
G0 TO 8
7 J0 = JJ - I(K1)
8 K = 2
K1 = L - M5
14 K2 = K1 + K1
IF(K2) 12, 12,
J0 = JJ - I(K1) - I(K2)
K1 = K1 + K1
M6 = M5
K = K + 2
G0 TO 14
12 IF(K2) 11,
J0 = JJ - I(K1) + 1
G0 TO 18
11 IF(K1) 18, 17
J0 = JJ + 1
G0 TO 18
17 J0 = JJ + I(K1)
18 J(L) = JJ + J(L)/M
IF(J(L)) 20, 20
J(L) = J(L) + M
20 CONTINUE
RETURN
END
SUBROUTINE FACTOR(SS, NN, JF, L, IPRT)

PURPOSE: FACTORIZATION ROUTINE.

ARGUMENTS: NUMBER TO BE FACTORIZED IS \( SS(1) \times 10^{(NN-1)} + \ldots + SS(NN) \).

IF \( NN \geq 3 \), HAVE \( SS(1) < 10^{**4} \). IF \( NN = 0 \), NUMBER TO BE FACTORIZED IS \( SS = SS(1) \). NUMBER MADE POSITIVE BEFORE BEING FACTORIZED AS \( \prod_{i=1}^{L/2} JF(2N-1) \times JF(2N) \). ATTEMPTS TO FACTOR \( SS \) USING PRIMES \(< 10^{**4} \) WHICH MUST BE AT BLOCK 1 ON TAPE 2.

IF IPRT NEQ 0, FACTORS ARE PRINTED -. * OR ** INDICATES A NUMBER WITH NO PRIME FACTOR \(< 10^{**4} \).

RESTRICTIONS: NUMBER TO BE FACTORIZED \(< 10^{**1000} \). \( NN = 0 \) OR \( 3 \leq NN \leq 250 \).

CALLS: IJMODK, IZR, PRIFAC

R-REGS: S56 - 10

IF NUMBER TO BE FACTORIZED \(< 10^{**100} \), HAVE DIM: JF(108) IN MAIN PROG
IF NUMBER TO BE FACTORIZED \(< 10^{**200} \), HAVE DIM: JF(186) IN MAIN PROG
IF NUMBER TO BE FACTORIZED \(< 10^{**300} \), HAVE DIM: JF(258) IN MAIN PROG
IF NUMBER TO BE FACTORIZED \(< 10^{**400} \), HAVE DIM: JF(326) IN MAIN PROG
IF NUMBER TO BE FACTORIZED \(< 10^{**500} \), HAVE DIM: JF(392) IN MAIN PROG
IF NUMBER TO BE FACTORIZED \(< 10^{**600} \), HAVE DIM: JF(456) IN MAIN PROG
IF NUMBER TO BE FACTORIZED \(< 10^{**700} \), HAVE DIM: JF(520) IN MAIN PROG
IF NUMBER TO BE FACTORIZED \(< 10^{**800} \), HAVE DIM: JF(582) IN MAIN PROG
IF NUMBER TO BE FACTORIZED \(< 10^{**900} \), HAVE DIM: JF(642) IN MAIN PROG
IF NUMBER TO BE FACTORIZED \(< 10^{**1000} \), HAVE DIM: JF(702) IN MAIN PROG

INTEGER SS(1), JF(1)
INTEGER J(1229), J1230, S(250), MM(250)
IF(NN) 1, 1
MM(1), III = 1
J1230 = 10007
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
LNSTAR, JPREV = 0
JdJ = 1
IF(NN) 26, 6
N = IABSF(SS(1))
IF(NN = 100140048) 9
L = L + 2
JF(L) = 1
GO TO 25
IF(N-2) 34
JF(L+1) = N
S(3) = N/100000000
S(2) = N/10000 . 1000 * S(3)
S(1) = IZR(M, N)
N = 3
GO TO 10.
18 15 IF(JPREV) 13, 13
JPREV = J(I)
 17 16 IF(JPREV) 13, 13
JPREV = J(I)
 16 15 IF(JPREV) 13, 13
JPREV = J(I)
 15 14 IF(JPREV) 13, 13
JPREV = J(I)
 14 13 IF(JPREV) 13, 13
JPREV = J(I)
 13 12 IF(JPREV) 13, 13
JPREV = J(I)
 12 11 IF(JPREV) 13, 13
JPREV = J(I)
 11 10 IF(JPREV) 13, 13
JPREV = J(I)
 10 9 IF(JPREV) 13, 13
JPREV = J(I)
 9 8 IF(JPREV) 13, 13
JPREV = J(I)
 8 7 IF(JPREV) 13, 13
JPREV = J(I)
 7 6 IF(JPREV) 13, 13
JPREV = J(I)
 6 5 IF(JPREV) 13, 13
JPREV = J(I)
 5 4 IF(JPREV) 13, 13
JPREV = J(I)
 4 3 IF(JPREV) 13, 13
JPREV = J(I)
 3 2 IF(JPREV) 13, 13
JPREV = J(I)
 2 1 IF(JPREV) 13, 13
JPREV = J(I)
 1 0 IF(JPREV) 13, 13
JPREV = J(I)
GO TO 19.
13 IF(J(I) = JPREV) 21,
  JF(L): = 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) = JPREV
  L = L + 2
  JF(L) = JJJ.
  JPREV = N.
24 JJJ = JJJ - 1.
20 IF(JPREV) 25,
  JF(L) = JPREV
  L = L + 2
  JF(L) = JJJ.
25 IF(IPRINT) 26,
  NCOL = 5
  CALL PRIFACT(JF,L,NCOL,NSAR)
  IF(NCOL = 150) 30,
  PRINT 31:(JF(1): I = 1,L)
31 FORMAT(5x1x,111,16)
  IF(NSAR) 32,
  PRINT 33:
33 FORMAT(3H %)
32 RETURN
30 CALL OUTREG:
26 RETURN
END
SUBROUTINE FINDRT(LQ, LPOWER, LNUM, LDEN, LROOT, LANS)

PURPOSE: CALCULATES LANS FROM LANS = LROOT \cdot LNUM / LDEN (MOD LQ**LPOWER)
          IF LANS NOT FOUND, LANS = 1

RESTRICTIONS: R*(LQ**LPOWER), LROOT \geq 2

CALLS: IZI, IZMR, IZR

B-REGS: 0-9

M O = LQ**LPOWER
.LX = IZMR(MQ, 0, IZI(LDEN, MQ), LNUM)
.LXX = IZR(LQ, LX)
DO 1 L! = 1, LQ=1.
.LX = LI

DO 2 LL: = 1, LROOT=1
.LX = IZMR(LQ, 0, LLY)
IF(LY = LXX), 4;
CONTINUE!

PRINT 5
FORMAT(9H FINDRT, 5X, 26H NO ROOT FOUND =, LANS = 1)
LANS: = 1:
RETURN

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
.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)

PURPOSE: IVAL = POWER OF LP LE MAX DIVIDING I. IF THIS IS MAX, IVAL = -1.

RESTRICTIONS: RMOD(LP*MAX,I)

CALLS IZR

B-REGS: 0 - 5

LD = 1;
DO 1: L = 1, MAX
LD = LP*LL.
LL = IZR(LL, I)
IF(LL) 2, 2
1. CONTINUE;
IVAL = -1.
RETURN
2. IVAL = L*1.
RETURN
END
FUNCTION IZI (IX, M)

PURPOSE: RECIPROCAL OF IX MOD M
ARGUMENTS: M IS MODULUS
RESTRICTIONS: R*{M}
CALLS IZI, IZQ, IZR
B-REGS, 0 - 5:

IF NO RECIPROCAL, IZI = 0.
IZI = IZI(IX, M) IS EQUIVALENT TO CALL RECIPR(IY, IX, M, 6)

DIMENSION I(40), K(40)
K1 = IX
K2 = M
K3 = K1/M = M
IF (K1), 9, 1:
K1 = K1 + M
1 L = 1
2 I(L) = IZQ(0, 0, 0, K1, K2)
   = K2 = K1 + I(L)
   IF (J), 3, 2
   L = L + 1
   K2 = K1
   K1 = J
GO TO 2
3 IF (K1 = 1), 5, 4
4 IZI = 0
RETURN
5 IF (L = 2), 6, 6
IZI = M - I(1)
RETURN
6 IF (L = 1), 7, 6
IZI = 1
RETURN
7 K(L) = 1
K(L = 1) = I(L = 1)
DO 8 LL = L, 2, 1
8 K(LL) = K(LL + 2) + K(LL + 1) + I(LL)
L = IZR(2, L)
IF (LL) 99, 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), IPRINT
C CALLS IZQ,IZR
C B-REGS: 0=6
C
C IF NO RECIPROCAL IZIU = HCF OF IX AND M:
C IX = IZIU(IX,M,IPRINT) IS EQUIVALENT TO CALL: RECIPR(IY,IX,M,IPRINT)

DIMENSION I(40),K(40)
K1=IX
K2=M
K3=K1=M
IF(K1),6,1
K1=K1+M
1 L=1
2 I(L) = IZQ(0,0,0,K1,K2)
   W=K2/K1*I(L)
   IF(J),3,7
   L=L+1
   K2=K1
   K1=W
   GO TO 2
3 IF(K1=1),5,1
4 IZIU = K1
PRINT 10,IX,M,K1
10 FORMAT(8H HCF OF ,12.5H AND ,12.4H IS ,12.10X,21H IZIU: SET EQUAL TO HCF)
   RETURN
5 IF(L=2) 6,6
6 IZIUU = M = I(1)
   IZIU = IZIUU
   GO TO 13
7 IF(L=1) 7,3
8 K1=1
9 IZIUU = M = K(1)
10 IZIU = IZIUU
11 IZIUU = IZIU
12 IF(IPRINT),11,13
13 PRINT 12,M,IX,IZIUU
14 FORMAT(8H MODULO ,12.5H RECIPROCAL OF ,12.4H IS ,12.3H (=IZIU))
FUNCTION IZMO(0,0,0,M,0,i,j)

PURPOSE: IZMO: SET TO QUOTIENT (i*j)/M.

ARGUMENTS: FIRST, SECOND, THIRD AND FIFTH ARGUMENTS ARE DHUMMY.

CONVENIENTLY 0.

REstrictions: R*(M) + RMOD(M,i,j)

B-REGs . 1:3.

OVERWRITES CALLING SEQUENCE.

WRITTEN IN ASP
FUNCTION: IZMR(K, I, J)

PURPOSE: IZMR: SET EQUAL TO I*J (MOD K)

ARGUMENTS: 0 ≤ IZMR < K, SECOND ARGUMENT IJ DUMMY, CONVENIENTLY ZERO

RESTRICTIONS: RMOD(K, I, J) RIF

B-REGS.: 1-3

OVERWRITES CALLING SEQUENCE
IDENTICAL TO IJMODK
WRITTEN IN ASP
FUNCTION IZMRU(M, I, J)

C PURPOSE: GIVES I*J MOD M
C
C RESTRICTIONS, M > 0
C
C CALLS IZMR, IZR
C
C B-REGS: 0 = 4

IE(I) 1,2
IZMRU = 0
RETURN
1 K = M-IZR(M, I)
IF(K < M) A,3,4
2 K = IZR(M, I)
4 IE(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 = 1**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
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
FUNCTION: IZQ(0, 0, 0, M, I)

PURPOSE: IZQ SET EQUAL TO I/M

ARGUMENTS. FIRST, SECOND AND THIRD ARGUMENTS DUMMY, CONVENIENTLY ZERO.

RESTRICTIONS. R+(M), R0(I), RIF

B-REGS. 1..3,124

OVERWRITES: CALLING SEQUENCE

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
FUNCTION IZR(M, I)
PURPOSE: IZR SET EQUAL TO I MOD M.

RESTRICTIONS: R*(M), R0(I), RIF.

B-REGS: 1 - 3, 124.

OVERWRITES CALLING SEQUENCE
WRITTEN IN ASP.
FUNCTION IZRU(M; I)

C PURPOSE: GIVES I MOD M
C
C RESTRICTIONS: R(M)
C
C CALLS IZR
C
B=REGS, 0 = 3:

.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)

PURPOSE: PLANTS J, \sqrt{J} = 1728 OR \sqrt{J} IN I, ACCORDING AS J123 = 1, 2 OR 3.

ARGUMENTS: I INTEGER ARRAY. N NUMBER OF TERMS COMPUTED. M IS MODULUS.
EG: IF J123 = 1, I(1) = 1, I(2) = 744 (MOD M), I(3) = 196884 (MOD M) .... I(0) OVERWRITTEN.

RESTRICTIONS: FOR CORRECT END AT TOP OF SERIES, MAKE N ZERO MOD J123.
M MUST NOT BE ZERO MOD 691. REUL(M, N) R+(M) K(N) M ^F.
WHERE K(N) = D(H) AND H IS LARGEST HIGHLY COMPOSITE NUMBER < N

CALLS: EUCLDIV, IZI, IZMR, IZR

B-REGS: 0 - 10

LIST OF HIGHLY COMPOSITE NUMBERS N

<table>
<thead>
<tr>
<th>N</th>
<th>D(N)</th>
<th>N</th>
<th>D(N)</th>
<th>N</th>
<th>D(N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>240</td>
<td>20</td>
<td>7560</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>360</td>
<td>24</td>
<td>10080</td>
<td>72</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>720</td>
<td>30</td>
<td>15120</td>
<td>80</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>840</td>
<td>32</td>
<td>20160</td>
<td>84</td>
</tr>
<tr>
<td>24</td>
<td>8</td>
<td>1260</td>
<td>36</td>
<td>25200</td>
<td>90</td>
</tr>
<tr>
<td>30</td>
<td>9</td>
<td>1680</td>
<td>40</td>
<td>27720</td>
<td>96</td>
</tr>
<tr>
<td>48</td>
<td>10</td>
<td>2520</td>
<td>48</td>
<td>45360</td>
<td>100</td>
</tr>
<tr>
<td>60</td>
<td>12</td>
<td>5040</td>
<td>60</td>
<td>50400</td>
<td>108</td>
</tr>
<tr>
<td>120</td>
<td>16</td>
<td>7560</td>
<td>80</td>
<td>75600</td>
<td>112</td>
</tr>
<tr>
<td>180</td>
<td>18</td>
<td>10080</td>
<td>84</td>
<td>10080</td>
<td>112</td>
</tr>
</tbody>
</table>

DIMENSION I(I)

DO 1 L = 1, N
   J(L) = 0
   IF(J123 = 2), 6, 7
   LC = IZMR(M, 0, 5520, IZI(691, M))
   LS1 = 10
   LEULER = 24
   GO TO 8
   LC = M = 504 * 504 / M * M
   LS1 = 4
   LEULER = 12
   GO TO 8
   LC = 240
   LS1 = 2
   LEULER = 8
   DO 99 LL = 1, N
   DO 100 LL = L, N, L
      LX = L
      DO 101 LS = 1, LS1
         EX = IZMR(M, 0, L, LX)
      100   J(LL) = I(LL) + LX
      CONTINUE
   DO 2 L = 1, N
      I(L) = IZMR(M, 0, LC, I(L))
   DO 3 L = 1, LEULER
      CALL EUCLDIV(I, I, N, M)
   99 CONTINUE
   101 CONTINUE

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

PURPOSE. COMPUTES ARRAYS XX = X, YY = Y FROM RELATIONS GIVEN BELOW.
ARGUMENTS, MM IS MODULUS,
Q INDEX (GAMMA Q) Q<51
MT GIVES XX,YY, UP TO AND INCL. TERM IN Z**MT.
D1/D2 ETC, ARE COEFFS (DENOM NEG IF NEC)
K*Q = K1/K2, (NUMERATOR NEG IF NEC)

KLN =: 1 FOR J
2 FOR SQRT(J=1728)
3 FOR CUBE ROOT J.

MM = MODULUS:
WHERE
Q EVEN N = (Q-4)/2 DDD1(1)/DDD2(1) = 1
Q ODD N = (Q-3)/2 DDD1(1)/DDD2(1) = 1; DDD1(1)/DDD2(1) = 0
RESTRICIONS. R*(MM), Q < 51
CALLS COEFJ, IMODK, KLEINJ, RECIPR
B=REGS, Q = 20
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(P0),X01(2000),D4(50),X02(2000),D5(50),X(8,2000),D6(50)
INTEGER A,AK,AAK,B,Q,Q6,RECIP,S,S1,S2,SS1,SS2
NFT = MT
N6 = NFT+2
Q6 = Q = 6
NN = (Q=3)/2
N6N = NT/Q+2
CALL RECIPR(INV2,2,MM,0)

ZEROISE ARRAYS.
DO 1000 I = 1,50
1000 D5(I),D6(I),D7(I),D8(I) = 0
DO 1001 I = 1,NFT
1001 Y1(I),Y2(I) = 0
DO 1002 I = 1,NN+2
1002 X(I,II),YX(I,II) = 0
1002 CONTINUE!

EVALUATE: C,CC,CCC
DO 1: I = 1,3
CALL RECIPR(Q,D2(4-I)*MM*MM,0)
C(I) = IMODK(MM,0,D1(4-I),II)
Q(4) = I:
DO 4 I = 1, NN+1
CALL: RECP1(I), DD2(NN+2 = 1) + MM, MM, 0
4 CC(I) = IJMODD(MM, 0, DD1(NN+2 = 1), II)
DO 5 I = 1, NN+3
CALL: RECP1(I) , DD2(NN+4 = I) + MM, MM, 0
5 CC(I) = IJMODD(MM, 0, DD1(NN+4 = I), II)

IF (Q = Q/2*2) 2, 2
SS1 = 0
CALL: RECP1(SS1, MM = (NN+2), MM, 0)
CC(NN+3) = 1  ! DATA SHOULD GIVE THIS
GO TO 3
2 CALL: RECP1(SS1, MM = 0, MM, 0)
SS2 = IJMODM(MM, 0, SS1, II)
CC(NN+1) = 1  ! DATA SHOULD GIVE THIS
CC(NN+3) = 0  ! DATA SHOULD GIVE THIS

SET UP J(KLN)
3 CALL: KLE1NJ((FC, (NN+1)*6*6+6, MM, KLN)
CALL: RECP1(II, K2*MM, MM, 0)
AK = IJMODD(MM, 0, K2, II)
CALL: RECP1(AAK, AK, MM, 0)
DO 6 I = 1, NNN
AAK = IJMODD(MM, 0, AAK, AK)
AA(K = (Q*(I-2)) = IJMODD(MM, 0, FC(I), AAK)
X01(0) = F(1)
X02(0) = CCC(1)
DO 7 I = 1, NNN+2:
X(I, = 2*I) = 1
DO 8 I = 1, NNN
YX(I, = 2*I) = 1
Y1(-3), Y2(-3) = 1
DO 9 N = -5*NTT - 3
CALL: COEFF(Y1, -3, Y1, = 3, N, Y2(N), MM)
DO 10 I = 2, N*4
210 WI(I) = X1(I, I)
CALL: COEFF(W, = 2, W, = 2, N*2, X(2, N+2), MM)
DO 10 I = 2, NNN+1
DO 211 L = 2*I, N+1
W(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(I, N+1), MM)
DO 11 I = 2, NNN
DO 212 L = 2*I, N+1
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
S = S + IJMODD(MM, 0, C(I+1), X(I, N))
S2 = X01(N) - S*Y2(N)*MM
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, CC(I+1), X(I, N1))

S2 = S * X02(N1) = AJ(N1)*MM

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

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

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 = 2, 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 ARGUMENTS. M IS MODULUS. N = NO. OF TERMS.
C RESTRICTIONS. R = (M). REULO(M,N,NN). J MUST NOT BE ARRAY.
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!
  K2 = K1 - K*MM
  IF(K2) 2,2,1
  JJ = JJ - I(K1) - I(K2)
  K1 = K1 - K*M6 - M5
  K = K + 2
  GO TO 4
2
  IF(K2) 1,,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!
  K2 = K1 - K*MM
  IF(K2) 12,12,14
  JJ = JJ + I(K1) + I(K2)
  K1 = K1 - K*M6 - M5
  K = K + 2
  GO TO 14
12
  IF(K2) 11,,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*MM
  IF(J(L)) 20,20,20
  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 FACTORISE IT.

C ARGUMENTS: FIRST DIGIT OF NUMBER IS ON CARD 1 COLUMN FCOL, LAST DIGIT OF NUMBER IS ON CARD NCARD, COLUMN FCOL. IF NPRINT NEG. 0 WILL PRINT NUMBER, FACTORS LEFT IN ARRAY JF(I), I = 1,LI AS IN FACTOR.

C RESTRICTIONS NUMBER MUST BE POSITIVE AND HAVE GREATER THAN 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,14,4Z,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
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:
S(NN) = S(MM+J)
CONTINUE!

NN DIGITS
NGRP4 = NN/4
NFRONT = NN/4*NGRP4
J = 1
IF(NFRONT) = 4,
DO 5 I = 1,4-NFRONT
J = J + 1.
5 S(J) = 0:
NGRP4 = NGRP4 + 1:
J = J + 4
DO 6 I = 1,NGRP4
J = J + 4
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 ID2Z(1)
CALL FACTORS(NGRP4,JF,L,1)
RETURN
END
SUBROUTINE SUMRES(I, IN, J, JN, IANS)

C PURPOSE: COMPUTES CHARACTER SUMS.
C ARGUMENTS, I, J, IANS INTEGER ARRAYS.
C CURVE IS
C \[ y^2 = J(1) x^{JN-1} + \ldots + J(JN) \]
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 RESPS. 0 - 10

DIMENSION I(1), J(1), IANS(1), K(100), KK(1000)

DO 1 L = 1, IN
  LR = I(L)
  KK(1) = 0
  DO 3 LL = 2, LP
  KK(LL) = -1
  DO 7 LLL = 1, (LP-1)/2
    LL = IZMR(LP, 0, LLL, LLL)
    KK(LL + 1) = 1
  DO 2 LL = 1, JN
    IF (J(LL)) 4, 4
    KL = LP + IZR(LP - J(LL))
    GO TO 2
  4 K(LL) = J(LL)
  CONTINUE
  KKK = 0
  DO 5 LL = 0, LP-1
    KL = K(1)
    DO 6 LLLL = 2, JN
      KL = IZMR(LP, 0, LLL, KL) + K(LLL)
    KL = IZR(LP, KL)
    KKK = KKK + KL
    6 CONTINUE
  5 IANS(L) = KKK
  RETURN
END
PURPOSE: MULTI-LENGTH ARITHMETIC PACKAGE.

CALLS /ML

B-REGS: 1, 56, 63, 65, 69, 81, 85, 91, 92, 97, 124

THESE ROUTINES ARE WRITTEN IN ASP.
SUBROUTINE PG(A,N,MM,HMN,X,P)

PURPOSE: COMPUTES DISCRIMINANT USING CHINESE REMAINDER THEOREM.

ARGUMENTS:
A INTEGER ARRAY, MM=VARIABLE X. SET TO DISCRIMINANT OF
A(1) X**N + ... + A(N+1) USING MM MODULUS MM(I), I = 1, MMM.

P IS: PRECISION.

RESTRICTIONS:
N < 26, R*(MM(I)), I = 1, MMM.

CALLS DETMOD, IJMODK, IZR, MLCREM

8-REGS: 0 - 12.

INTEGER P,B(49,49),A(1),MM(1),DET(50),AAO(26),AA(26),AA1(26)
N1 = N + 1
N2 = 2*N - 1
DO 5 I = 1,26
  AA0(I),AA(I),AA1(I) = 0
DO 1 L = 1,MMM
  M = MM(L)
  DO 2 I = 1,N1
    IF(A(I)) = 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
    B(I,J) = AA(J,K)
    CONTINUE!
  6 AA(N1) = 0
  DO 7 I = 1,N1
    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
        B(I,J) = AA(J,K)
        CONTINUE!
      9 B(I,1) = N
      B(I,2) = 1.
CALL: DETMOD(B,49,N2,DET(L),M)
CALL: MLCREM(DET,MM,MMM,X,P)
RETURN
END
SUBROUTINE MLCF(X, Y, P)

PURPOSE: OUTPUTS CONTINUED FRACTION.

ARGUMENTS: X, Y, ML-VARIABLES. P PRECISION. OUTPUTS CONTINUED FRACTION OF
X UNTIL CORRESPONDING PARTIAL QUOTIENTS OF X AND Y DIFFER.
NORMALLY: Y = X * 10**(P+5), SAY.

CALLS MLC.
8-REGS. 0 = 4.

EXTERNAL! IPART, FPART, RCP
C 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 ML COS(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 F FROM CURRENT INPUT STREAM ON FIRST EXECUTION:

INTEGER P
EXTERNAL READ, SQ, POS
IB(I,II)1,1:
IIV = 1
CALL ML DLK(P), NAMES(EPS, PI, F, T))
CALL ML D(P), READ, TO(PI), PI, ADD(PI), TO(PI), S(10), POWER(-P), 1 TO(EPS))
CALL ML<TO(PI), MODULO(PI), TO(F), S(0), TO(Y), S(1), TO(T))
N = 1:
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)

PURPOSE: SOLVES SYSTEM OF CONGRUENCES IANS = \sum_{L=1}^{N} \left( \frac{A(L)}{LP(L)} \right) \mod \prod_{L=1}^{N} LP(L)

ARGUMENTS: I, LP INTEGER ARRAYS. IANS ML-VARIABLE. ELEMENTS OF LP MUST BE RELATIVELY PRIME IN PAIRS.

RESTRICTIONS: 2 \leq N \leq 50, R(LP(L)), L = 1 \ldots N

CALLS IZI, IZMR, ML

8-REGS: 0 \ldots 9

FORMS: PROD = \prod_{L=1}^{N} LP(L) AND PROD/LP(1) ONCE AND FOR ALL P. MUST BE LARGE ENOUGH TO ENABLE N*PROD*MAX(LP(L)) TO BE CALCULATED.

IANS COMPUTED MODULO PROD. IF GREATER THAN PROD/2, IANS MADE NEGATIVE.

DIMENSION I(1), LP(1), A(50)
CALL MLO(PREC(P), SL(0), TO(IANS))
ML(P) 4, 4
IJI = 1:
CALL MLD(PREC(P), NAMES(PROD, PRODH))
DO 6, LL = 1 \ldots N
CALL MLD(PREC(P), NAMES(A(LL)))
CALL MLO(PREC(P), SL(1), TO(PROD))
DO 1, L = 1 \ldots N
CALL MLO(PROD, MULT(SL(LP(L))), TO(PROD))
DO 2, L = 1 \ldots N
LIP = LP(L)
J = 1
DO 3, LL = 1 \ldots N
LIP = LP(LL)
IF (LIP = LLP) 3,
J = IZMR (LP, 0, J, LLP)
CONTINUE
2 CALL MLO(PROD, DIV(SL(LLP)), MULT(SL(IJI(LLP))), TO(A(LL)))
DO 5, L = 1 \ldots N
5 CALL MLA(A(L), MULT(SL(j(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)

PURPOSE:
ML VERSION OF CONVRA

ARGUMENTS:
X ML VARIABLE, I, J, K ML-ARRAYS, P PRECISION,

ON RETURN I CONTAINS PARTIAL QUOTIENTS, J, K NUMERATOR,
DENOMINATOR OF CONVERGENTS TO X, PRINTER THESE, COMPUTES UNTIL,
NUMERATOR + DENOMINATOR $> 10^N$ OR UNTIL LTEST PARTIAL QUOTIENTS
HAVE BEEN COMPUTED, WHEN ASTERISKS ARE PRINTED, LOUT = NUMBER
OF CONVERGENTS COMPUTED

CALLS ML

B-REGS: 0 = 10

EXTERNAL RCP, IPART, POS

DIMENSION I(1), J(1), K(1)

IF(I1) 1 1,
1
IF(I0) 1 1,
1
CALL: MLO(PREC(P), NAMES(Y, Z, LIMIT, RECIP))
CALL: MLO(PREC(P), SL(I0), 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(I0), TEXT(14H CONVERGENTS OF), X, PRINT(10, P))
L = 3
5
CALL: MLO(Y, IPART, TO(J(I0)))
CALL: MLO(I(I0), MULT(J(L - 1)), ADD(J(L - 2)), TO(J(I0)))
CALL: MLO(I(I0), MULT(K(L - 1)), ADD(K(L - 2)), TO(K(I0)))
CALL: MLO(PREC(P), NL(1), SL(L - 3), PRINT(5, 0), I(I0), PRINT(15, 0), J(I0))
1
PRINT(90, 0), K(I0), PRINT(40, 0)
IF(MLT(J(I0)), ADD(K(I0)), SUB(LIMIT))) 4 2 2:
4
CALL: MLO(Y, SUB(I(I0))), 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 MUDC(A, N, X, P)

PURPOSE: Computes discriminant of polynomial with multi-length coefficients.

ARGUMENTS: A is ML-ARRAY, X ML-VARIABLE, P PRECISION. X SET TO

RESTRICTIONS, N < 26 AND SAME FOR ALL CALLS IN ONE RUN:

CALLS MLD, ML!

B-REGS, 0 - 9.

INTEGER P
DIMENSION B(49, 49), BB(49, 49), A(1), AA(26), AA(26), AA1(26)

IF(I,I), I = 1
II = N - 1
N2 = 2*N! - 1
DO 5: I = 2, N, N2 + 1
CALL MLD(PREC(P), NAMES(AA(I)))
DO 5: I = 1, N2
DO 50 J = 1, N2
CALL MLD(PREC(P), NAMES(B(I, J), BB(I, J)))
CONTINUE!
DO 2: I = 1, N1
CALL MLO(A(I), TO(AA(I)))
K = 1
DO 4: I = 2, N2, 2:
K = K - 1
DO 6 J = 1, N2
CALL MLO(AA(J, K), TO(B(I, J)))
CONTINUE!
CALL MLO(PREC(P), SL(0), TO(AA(N1)))
DO 7 I = 1, N1
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
CALL MLO(AA(J, K), TO(B(I, J)))
CONTINUE!
CALL MLO(PREC(P), SL(N), TO(B(1, 1)), SL(1), TO(B(2, 1)))
CALL MLD(B, BB, X, N2, P, 49)
RETURN.
END
SUBROUTINE MLDEBT(B, A, DET, M, P, IA)

C PURPOSE: ML VERSION OF M0Q2AM.

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(lA, 1), A(lA, 1), D(20), IND(20), JND(20)

IF(lA) 1, 1
lA = 1
CALL! MLD(PREC(P), NAMES(AMAX, STO))

DO 101 l = 1, 20
101 CALL: MLD(PREC(P), NAMES(D(l)))

1 CALL! MLO(PREC(P), SL(0), TO(AMAX))

DO 2: I = 1, M

IND(I) = 1
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))

4 I4 = I
J4 = J

7 CONTINUE!
2 CONTINUE!

DO 111: J = 1, MM

IF(l4-J) 6, 6, 4

DB = 1
MN = M-1

DO 111: J = 1, MM

IF(l4-J) 6, 6, 4

DB = DD
ISTO = IND(J)
IND(J) = IND(I4)

IND(I4) = ISTO

DO 5: K = 1, M

CALL! MLO(A(I4, K), TO(STO), A(J, K), TO(A(I4, K)), STO, TO(A(J, K)))

6 IF(J4-K) 8, 8, 9

DB = DD
ISTO = JND(J)
JND(J) = JND(I4)
JND(I4) = 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))

11 J1 = J

DO 11 I = 1, M

CALL! MLD(PREC(P), NAMES(AMAX, STO))

END
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 ML'EXP(X,Y,P)

PURPOSE: COMPUTES ML EXPONENTIAL ARGUMENTS, X, Y ML-VARIABLES, P PRECISION, SETS Y = EXP(X)

CALLS ML
B-REGS: 0 - 4

COMPUTES E = EXP(1) ONCE AND FOR ALL ON FIRST EXECUTION.

EXTERNAL IPART, FPART, POS

IF(I!1) 2, 2:
I! = 1:
CALL MLO(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
N = N + 1
IF(MLT(E, ADD(F), TO(E), F, DIV(SL(N)), TO(F), SUB(EPS)))
2 CALL MLO(X, IPART, TOSL(1), X, FPART, TO(F), TO(T), SL(1), TO(Y))
N = 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(1), MULT(Y), TO(Y))
RETURN
END
SUBROUTINE: MLEXPO(X, Y, Z, P)

• PURPOSE: COMPUTES ML EXPONENTIAL
• ARGUMENTS: X, Y, Z ML-VARIABLES, P PRECISION. SETS Z = X**Y.
• CALLS MLEXP, MLLOG, ML
• B-REGS, Q = 5

IF(I1) 1, 1
I1 = 1
CALL: MLLOG(P), NAMES(A)
CALL: MLEXP(A, Z, P)
RETURN
END
SUBROUTINE MLJ(C,D; AA, BB, N, NN, NSPEC, P)

PURPOSE: COMPUTES KLEIN'S INVARIANT J (SMALL J)

ARGUMENTS: C, D, AA, BB ARE ML-VARIABLES, P PRECISION.

IF NSPEC NEQ 0, C AND NN ARE IGNORED AND AA IS SET TO:
RE: J(1/2 + D, I), BB TO IM J(1/2 + D, I), USING N TERMS OF J = 1, 744, 19684, ...
(MLCOS NOT ENTERED).

IF NSPEC = 0, AA IS SET TO RE J(C + D, I), BB TO IM J(C + D, I),
USING N TERMS OF EXPANSION. IF BOTH C AND D ARE OF FORM
C/INT, D/INT, WHERE INT IS AN EVEN INTEGER, SET NN = INT.
OTHERWISE SET NN = 0.

RESTRICTIONS: 3 ≤ N ≤ 400.
CALLS MLCOS, MLEXP, ML
B-REGS: 0 = 11

ON FIRST EXECUTION READS C FROM CURRENT INPUT STREAM AND READS:
C(1) = 1, C(2) = 744, ..., C(N) FROM INPUT STREAM 1; SELECTING INPUT
STREAM 0 AFTERWARDS

DIMENSION AJ(400)
INTEGER P
EXTERNAL READ, RCP, SQROOT, SQ, COMP, POS
IF (III) 4, 1
III = 1
PP = 2, 1, 3.14159265359
CALL MLR(PREC(P), NAMES(P, X, Y, Z, W, V, A, B, EPS, S, SS))
CALL MLR(PREC(P), READ, TO(P), PI, ADD(P), TO(P), SLK(10), POWER(P))
1 TO (EPS))
CALL INPSEL(1)
DO 2: I = 1, N
CALL MLR(PREC(P), NAMES(AJ(I)))
2: CALL MLR(PREC(P), READ, TO(AJ(I)))
CALL INPSEL(0)
1 IF (NSPEC) 9,
CALL MLR(D, MULT(P), TO(A))
CALL MLEXP(A, S, P)
CALL MLR(S, COMP, TO(B), RCP, TO(A), B, ADD(AJ(2), TO(S), SLK(1), TO(B))
DO 10, I = 3, N
IF (MLT(B, MULT(A), TO(B), MULT(AJ(I)), TO(SS), ADD(S), TO(SS), SS, POS, 1)
SUB(EPS)) 11, 11;
10 CONTINUE
11 CALL MLR(S, TO(AA), SL(0), TO(BB))
RETURN
9 CALL MLR(C, TOSL(XA1), MULT(P), TO(A), COMP, TO(Z), D, TOSL(B1), MULT(P),
1 TO(B))
Z1 = PP*A1
Z2 = Z1
NNN = NN/2
CALL MLEXP(B, X, P)
CALL MLR(X, RCP, TO(Y))
CALL MLCOS(Z, W, P)
CALL: MLO(AJ(4), MjLT(X), TO(V), MULT(W), TO(AA))
CALL: MLO(W, SQ, COMP, ADD(SL(1)), SQ ROOT, TO(W))
IF(SINF(22)), 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
   W = J + 1
   CALL: MLO(X, MjLT(Y), TO(X), Z, ADD(A), TO(Z))
   Z2 = Z2 * Z1:
5
CALL: MlCOS(Z, WP)
6
CALL: MLO(AJ(I), MjLT(X), TO(V), MULT(W), ADD(AA), TO(AA))
IF(J = K) 8, 8
7
K = K + NNN
8
GO TO 4
CALL: MLO(W, SQ, COMP, ADD(SL(1)), SQ ROOT, TO(W))
9
IF(SINF(22)), 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 (I) 2, 2
I0 = 1
CALL MLX(PREC(P)!, NAMES(Z, A, B, C, EPS))
CALL MLX(PREC(P)!, SL(10)!, POWER(-P)!, TO(EPS))
CALL MLX(X, TOSL(XX))
XX = LOGF(XX)
CALL MLX(PREC(P)!, SL(XX)!, TO(Y))
CALL MLX(Y, Z!, P)
CALL MLX(X, DIV(Z)!, SUB(SL(1)!, TO(Z)!, TO(B)!, TO(A))
N = 1
CALL MLX(A, MULT(B)!, TO(A)!, DIV(SL(-N)!, ADD(Z)!, TO(Z))
N = N + 1
CALL MLX(A, MULT(B)!, TO(A)!, DIV(SL(N)!, TO(C)!, ADD(Z)!, TO(Z))
IF (MLT(C, POS, SUB(EPS)!) .GT. 1)
CALL MLX(Y, ADD(Z)!, TO(Y))
RETURN
END
SUBROUTINE MLPOW(A,N1,N2,Y,XX,P)

PURPOSE: ML EXPONENTIATION WITH RATIONAL EXPONENT, USING: NEWTON ITERATION.

ARGUMENTS: A, Y, XX ML-VARIABLES, SETS Y = A**(N1/N2), CALCULATED WITH
PRECISION P. XX MUST INITIALLY CONTAIN APPROXIMATION TO Y.

CALLS ML:

B-REGS: 0 - 7:

EXTERNAL RCP, POS
INTEGER P
IF (III) 1, 1

1 I = 1
CALL MLD(PREC(P),NAMES(EPS,B,R,XX,XXX))
CALL MLD(PREC(P),SL(10),POWER(-P*2),TO(EPS))

1 NN = N2 - 1
CALL MLD(A,POWER(N1),DIV(SL(N2)),TO(B),XX,TO(X),SL(NN),DIV(SL(N2))

2 CALL MLD(X,TO(XXX))
CALL MLD(X,POWER(NN),RCP,MULT(B),ADD(X,MULT(R)),TO(X))
IF (MLT(X, SUB(XXX), POS, SUB(EPS))) 2
CALL MLD(X,TO(Y))
RETURN
END
SUBROUTINE MLPOW3(A, Y, XX, P)

C PURPOSE: COMPUTES CUBE ROOT USING NEWTON ITERATION.
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
  I = 1
CALL MLO(PREC(P), NAMES(EPS, B, R, X, XXX))
CALL MLO(PREC(P), SL(10), POWER(-P+3), TO(EPS))
CALL MLO(A, DIV(SL(3), TO(B), SL(XX), TO(X), SL(2), DIV(SL(3), TO(R))
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)

PURPOSE: COMPUTES ROOT OF ALGEBRAIC EQUATION.

ARGUMENTS: AA ML-ARRAY, ROOT ML-VARIABLE, EQUATION IS: AA(1) x**NN + ... + AA(50) x + XINIT IS SINGLE LENGTH APPROXIMATION TO ROOT.

RESTRICTIONS: NN < 50

CALLS ML

B-REGS: 0 - 8:

DIMENSION AA(1), A(50)

INTEGER P, POWR
EXTERNAL COMP
IE(I) = 1
CALL MLD(PREC(P), NAMES(X, Y, F, G, DEL))

POWR = 0.4343*LOGF(ABSF(XINIT)+1)-P
N = NN
DO 1 I = 1, N + 1
1   AA(I) = AA(I)
   CALL MLD(PREC(P), SL(XINIT), TO(X), SL(10), POWER(POWR), TO(DEL))
   CALL MLD(PREC(P), SL(A(1)), TO(F))
   CALL MLD(PREC(P), SL(A(1)), TO(G))
   DO 2 I = 2, N + 1
2      CALL MLO(MULT(X), ADD(SL(A(I))), TO(F))
   CALL MLO(MULT(X), ADD(SL(A(I))), TO(G))
   IF (MLT(F)) 102, 199, 101
   IF (MLT(G)) 199, 199, 103
   IF (MLT(G)) 104, 199, 199
   IF (MLT(F, SUB(G))) 106, 204, 107
   IF (MLT(G, SUB(F))) 106, 204, 107
   CALL MLO(DEL, COMP, TO(DEL))
   DO 3 I = 1, N
3      CALL MLO(DEL, MUL(F), DIV(F, SUB(G)), ADD(X), TO(X))
   CALL MLD(PREC(P), SL(A(1)), TO(F))
   DO 4 I = 1, N + 1
4      CALL MLO(MULT(X), ADD(SL(A(I))), TO(F))
   IF (MLT(F)) 109, 100
   IF (MLT(G)) 111, 204, 100
   IF (MLT(G)) 204, 111
   CALL MLO(DEL, COMP, TO(DEL))
   GO TO 100
100 CALL MLO(X, TO(ROOT))
   RETURN
101 CALL MLD(PREC(P), SL(XINIT), TO(ROOT), NL(1), TEXT(21H*** NO: ROOT FOUND.
   RETURN
END
SUBROUTINE ML SIN(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: Q = 4
C READS: T FROM CURRENT INPUT STREAM ON FIRST EXECUTION.

INTEGER P
EXTERNAL READ, SQ, POS
IF(I31)1,1
I = 1
1 CALL ML-M(PRE(S), NAM (EPS, PI, F, T))
CALL ML-M(PRE(S), READ, TO(PI), PI, ADD(P1), TO(PI), SL(10), POWER(-P),
1 TO(EPS))
1 CALL ML-M(X, MODLO(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, SUBK
1 EPS))) 2:
RETURN
END