Extended precision computation of the incomplete beta function

Leigh Allen Ihnen
Iowa State University

Extended precision computation of the incomplete beta function

by

Leigh Allen Ihnen

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Major: Statistics

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa

1983

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

The incomplete beta function is important for finding probabilities of test statistics. Through the use of a transformation, the beta distribution will yield F or Student-T probabilities. This thesis presents and describes a program for the computation of the incomplete beta function and a new method for computing the inverse incomplete beta function. The main concern in developing the program to compute the incomplete beta function is to provide probabilities with a specified number of correct significant digits independent of the parameters. A secondary concern is to optimize the execution time subject to desired accuracy.

Existing algorithms are limited by the maximum machine precision on which they are run, the range of parameters and the accuracy of the result being machine dependent. To eliminate restrictions placed on accuracy due to the parameters, a package of machine independent multiple precision subroutines has been developed and programmed. Using these multiple precision subroutines, a program to compute the incomplete beta function with up to 70 correct significant digits has been implemented. The range of allowable parameters includes all numbers representable in IBM double precision, as well as, multiple precision numbers containing up to 70 decimal digits and exponents less than 50 in absolute value. Simple modifications of four subroutines will allow this program to run on any machine in which an integer word contains 32 or more bits.

1.2 Organization

Chapters 2, 3, and 4 contain background material needed to develop and implement an algorithm that evaluates the incomplete beta function. Chapter 5 describes the algorithm developed and implemented to compute the incomplete beta function. Existing algorithms for computing the inverse incomplete beta function and a new algorithm for the inverse incomplete beta function are discussed in chapter six. In Appendix A, the program computing the incomplete beta function is presented. A double precision program computing the inverse incomplete beta function is given in Appendix B.

Chapter 2 contains material on continued fractions. The definition of a continued fraction is provided and the general representation of a continued fraction that is used in this thesis is given. Also discussed in this chapter are properties of some continued fractions and different methods for evaluating a continued fraction.

In chapter 3, the beta distribution is described. The shape that the distribution takes for different parameter combinations is examined. The important relationships with other distributions are given, as well as the derivations of some power series representations of the incomplete beta function. Finally, existing programs for the evaluation of the distribution function are given and discussed.

Chapter 4 contains a list to desirable properties for a multiple precision (MP) package. A representation for a MP number and arithmetic operations on MP numbers is given. To evaluate the incomplete beta function several elementary functions are needed. A description of the algorithms chosen to evaluate several simple functions is given and the reason for the choice of each algorithm is given.

Chapter 5 uses the material given in chapters 2, 3, and 4 to develop an algorithm for the evaluation of the incomplete beta function. The algorithm must provide desired accuracy, which involves selecting methods of solution for the incomplete beta distribution with computable error bounds. Various methods are examined and considerations about speed of execution are discussed. Chapter 5 also contains computational considerations used to develop the program given in Appendix A, which evaluates the incomplete beta function. The last section of chapter 5 examines the number of MP digits needed to find a solution with sufficient significant digits.

2. CONTINUED FRACTIONS

Continued fractions provide alternate ways of computing power series. A given power series can have several related continued fraction representations, that might behave differently. For example, convergence of a power series does not necessarily imply convergence of a related continued fraction. Likewise, convergence of a continued fraction does not necessarily imply convergence of a related power series. In this chapter, some important results about continued fractions will be given. The material presented in this chapter is based on results given by Khovanskii (1956), Jones and Thron (1980), and Wall (1948).

2.1 Definitions

A continued fraction has the following general form.

$$b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \ddots \cfrac{a_r}{b_r + \ddots}}}}$$

An alternate representation is $[b_0, \frac{a_r}{b_r}]_1^\infty$, where a_n is the nth partial numerator, b_n the nth partial denominator, and $\frac{a_n}{b_n}$ the nth partial quotient. In the remainder of this thesis, continued fractions will be written as

$$b_0 + \frac{a_1}{b_1} + \frac{a_2}{b_2} + \dots \frac{a_r}{b_r} + \dots$$

The n th convergent is defined as $[b_0, \frac{a_r}{b_r}]_1^n$ or $b_0 + \frac{a_1}{b_1} + \frac{a_2}{b_2} + \dots \frac{a_n}{b_n}$.

Let $P_{-1} = 1$, $P_0 = b_0$, $Q_{-1} = 0$, and $Q_0 = 1$. Using the recurrence relations

$$P_n = b_n P_{n-1} + a_n P_{n-2} \tag{2.1}$$

$$Q_n = b_n Q_{n-1} + a_n Q_{n-2} \tag{2.2}$$

the n th convergent is $\frac{P_n}{Q_n}$.

The difference between two consecutive convergents is expressed below.

$$\frac{P_n}{Q_n} - \frac{P_{n-1}}{Q_{n-1}} = (-1)^{n+1} \frac{a_1 a_2 \cdots a_n}{Q_{n-1}\, Q_{n-2}} \tag{2.3}$$

Using (2.3), the differences between two consecutive even convergents or odd convergents can be expressed in the form

$$\frac{P_{2k+1}}{Q_{2k+1}} - \frac{P_{2k-1}}{Q_{2k-2}} = - \frac{a_1 a_2 \cdots a_{2k}\, b_{2k+1}}{Q_{2k-1}\, Q_{2k+2}} \tag{2.4}$$

$$\frac{P_{2k}}{Q_{2k}} - \frac{P_{2k-2}}{Q_{2k-2}} = \frac{a_1 a_2 \cdots a_{2k-1}\, b_{2k}}{Q_{2k-2}\, Q_{2k-2}} \quad . \tag{2.5}$$

Equations (2.4) and (2.5) are used to obtain:

$$\frac{P_{2k+1}}{Q_{2k+1}} = b_0 + \frac{a_1}{b_1} - \frac{a_1 a_2 a_2}{Q_1 Q_2} - \dots \frac{a_1 a_2 \cdots a_{2k}\, b_{2k+1}}{Q_{2k-1}\, Q_{2k+1}} \qquad (2.6)$$

and

$$\frac{P_{2k}}{Q_{2k}} = b_0 + \frac{a_1 b_2}{Q_0 Q_2} + \frac{a_1 a_2 a_3 b_4}{Q_2 Q_4} + \dots \frac{a_1 a_2 \cdots a_{2k-1}\, b_{2k}}{Q_{2k-2}\, Q_{2k}} . \qquad (2.7)$$

Suppose a given continued fraction converges and $b_n > 0$, $a_n > 0$ for $n \geq 1$. Then the even convergents form a monotonically increasing sequence of lower bounds for the quotient (2.7). Similarly, the odd convergents form a monotonically decreasing sequence of upper bounds for the quotient (2.6). These results are useful, because they provide an error bound, if the n^{th} convergent is to be used to approximate the continued fraction.

As more than one continued fraction exists for a given power series, there are several terms used to describe the relationship between a power series and a continued fraction. Let the power series be given as $\sum_{k=0}^{\infty} c_k x^k$. A continued fraction is said to be <u>equivalent</u> if the n^{th} convergent is equal to the sum of the first n terms in the power series. A <u>corresponding</u> continued fraction is one for which the expansion of the n^{th} convergent, by (2.6) or (2.7) into a power series, has coefficients identical to the coefficients of the power series up to the term containing x^n . A continued fraction, for which the power series expansion has coinciding coefficients for terms up to x^{2n} , is called an <u>associated</u> continued fraction.

2.2 Evaluation of Continued Fractions

Four general methods for evaluating a continued fraction exist. Two of the methods use a forward evaluation in which the number of convergents used to approximate the continued fraction is not fixed. The other two methods require that the nth convergent approximate the continued fraction within an acceptable amount of error. All four algorithms are described by Miklosko (1977). The fastest two algorithms are the methods that do not use forward evaluation. However, if it is not known that the nth convergent will supply sufficient accuracy, repeated evaluation of convergents by these two methods will be extremely slow. Hence, only the two forward methods will be given in this section.

To describe the first method, let

$$f_1 = a_1 \ , \quad q_1 = \frac{a_1}{b_1} \ ,$$

$$c_r = \frac{b_r}{f_{r-1}} \ , \quad f_r = a_r + c_r \ , \quad q_r = \frac{-c_r q_{r-1}}{f_r} \ ,$$

and denote the nth convergent as w_n . Then $w_1 = q_1$ and

$$w_n = w_{n-1} + q_n \quad \text{for} \quad n > 1 \ .$$

The other method for evaluation makes use of equations (2.1) and (2.2) and the definition of the nth convergent. This method computes the nth convergent w_n as

$$w_n = \frac{P_n}{Q_n} = \frac{b_n P_{n-1} + a_n P_{n-2}}{b_n Q_{n-1} + a_n Q_{n-2}}$$

where $P_{-1} = 1$, $P_0 = b_0$, $Q_{-1} = 0$, and $Q_0 = 1$.

3. BETA DISTRIBUTION

3.1 Parameters and Shape

The density function of the beta distribution is of the form

$$y = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} x^{p-1}(1-x)^{q-1} ,$$

where $p > 0$, $q > 0$, $0 \leq x \leq 1$, and $\Gamma(z)$ is the complete gamma function. Hence, the cumulative distribution function is

$$I_x(p,q) = \frac{B_x(p,q)}{B(p,q)} = \frac{1}{B(p,q)} \int_0^x t^{p-1}(1-t)^{q-1}dt ,$$

with $B(p,q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)}$. $I_x(p,q)$ is called the Incomplete Beta Function.

The shape of the beta distribution is determined by the parameters p and q . For $p < 1$ and $q < 1$, the distribution is U-shaped, with $y = \infty$ at $x = 0$ and $x = 1$. When $p < 1$ and $q > 1$, the distribution has a decreasing J-shape and $y = \infty$ for $x = 0$. If $p > 1$ and $q < 1$, the distribution is J-shaped increasing with $y = \infty$ for $x = 1$. The beta distribution becomes a uniform [0,1] when $p = 1$ and $q = 1$. When $p > 1$ and $q > 1$, the distribution is unimodal. The mode is $\frac{(p-1)}{p+q-2}$. The mean of the distribution is $\frac{p}{p+q}$ with the variance being $\frac{pq}{(p+q)^2(p+q+1)}$.

As either parameter or both parameters increase, the distribution begins to look like a single mass located at the mode. The graph of the distribution behaves like x^{p-1} , as x approaches 0 for $p > 1$. When $q > 1$, the graph behaves like $(1-x)^{q-1}$ as x nears 1 . For p equal q , the graph is symmetric around the mode. The degree of skewness is determined by the ratio of p to q . With p greater than q , as p increases the graph becomes more skewed to the right. Likewise, if q is greater than p , as q increases the graph becomes skewed to the left.

3.2 Related Distributions and Functions

For p equal to one and q equal to one, the beta distribution becomes the uniform [0,1] . Other relationships follow:

Binomial distribution

$$\sum_{j=p}^{n} \binom{n}{j} x^j (1-x)^{n-j} = I_x(p,n-p+1) .$$

Hence,

$$\binom{n}{p} x^p (1-x)^{n-p} = I_x(p,n-p+1) - I_x(p+1,n-p) .$$

Hypergeometric function

$$\frac{1}{B(p,q)} \frac{x^p}{p} F(p,1-q,p+1,x) = I_x(p,q)$$

Negative binomial

$$\sum_{j=p}^{\infty} \binom{n+j-1}{j} x^j (1-x)^n = I_x(p,n)$$

Students-t distribution

$$[1 - T(t,n)] = I_x(\tfrac{n}{2}, \tfrac{1}{2}) ,$$

where $x = \dfrac{n}{(n+t^2)}$

F distribution

$$[1 - F(y,n_1,n_2)] = I_x(\tfrac{n_2}{2}, \tfrac{n_1}{2}) ,$$

where $x = \dfrac{n_2}{n_2+n_1 y}$

3.3 Recurrence Relations

Soper (1921) gives eight recurrence relations for unit changes in the parameters. The recurrence relations allow p and q to be changed as follows, raise p and q, decrease p and q, decrease p with q constant, decrease q with p constant, raise p with q constant, raise q with p constant, raise q and decrease p, raise p and decrease q. Another important relationship of the incomplete beta involves the interchange of p and q.

$$I_x(p,q) = 1 - I_{1-x}(q,p) \tag{3.1}$$

With use of equation (3.1), the two recurrence relations, raise p and decrease q and raise p with q constant have been used to compute the incomplete beta function.

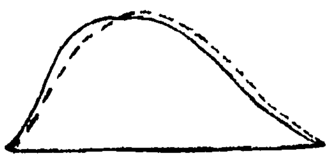
To derive the formula for a unit increase in p and a unit decrease in q consider the derivative

$$\frac{d}{dx}[x^p(1-x)^{q-1}] = px^{p-1}(1-x)^{q-1} - (q-1)x^p(1-x)^{q-2} . \qquad (3.2)$$

Integration of (3.2), followed by division by $B(p,q)$ yields

$$I_x(p,q) = \frac{\Gamma(p+q)}{\Gamma(p+1)\Gamma(q)} x^p(1-x)^{q-1} + I_x(p+1,q-1) . \qquad (3.3)$$

Soper refers to this process as reduction by parts. To see the reduction, consider the graph of the density function.



The solid line is $y = \frac{1}{B(p,q)} x^{p-1}(1-x)^{q-1}$, while the broken line represents $y' = \frac{1}{B(p+1,q-1)} x^p(1-x)^{q-2}$. Hence, $y' = y(\frac{q}{p})(\frac{x}{1-x})$. Since the area under both curves is one, $I_x(p+1,q-1) < I_x(p,q)$ with the restriction $q-1 \geq 0$.

The derivation of the raise p with q constant formula begins with

$$\frac{d}{dx} [x^p(1-x)^q] = px^{p-1}(1-x)^{q-1} - (p+q)\ x^p\ (1-x)^{q-1} . \qquad (3.4)$$

As before, integration of (3.4) followed by division by B(p,q) results in

$$I_x(p,q) = \frac{\Gamma(p+q)}{\Gamma(p+1)\Gamma(q)} x^p(1-x)^q + I_x(p+1,q) \ . \qquad (3.5)$$

In this case $y = \frac{1}{B(p,q)} x^{p-1}(1-x)^{q-1}$ and $y' = \frac{1}{B(p+1,q)} x^p(1-x)^{q-1}$.

Hence, $y' = y(\frac{p+q}{p})x$ and $I_x(p+1,q) < I_x(p,q)$.

After s applications of equation (3.3) ,

$$I_x(p,q) = \frac{\Gamma(p+q)}{\Gamma(p+1)\Gamma(q)} x^p(1-x)^{q-1}\{1 + \frac{q-1}{p+1}(\frac{x}{1-x})$$

$$+ \frac{(q-1)(q-2)}{(p+1)(p+2)}(\frac{x}{1-x})^2$$

$$+ \dots + \frac{(q-1)(q-2)\ \dots\ (q-s+1)}{(p+1)(p+2)\ \dots\ (p+s-1)}(\frac{x}{1-x})^{s-1}\}$$

$$+ I_x(p+s,q-s) \ . \qquad (3.6)$$

Similarly, after s applications of equation (3.5) ,

$$I_x(p,q) = \frac{\Gamma(p+q)}{\Gamma(p+1)\Gamma(q)} x^p(1-x)^q\{1 + \frac{p+q}{p+1} x + \frac{(p+q)(p+q+1)}{(p+1)(p+2)} x^2$$

$$+ \dots + \frac{(p+q)(p+q+1)\ \dots\ (p+q+s-2)}{(p+1)(p+2)\ \dots\ (p+s-1)} x^{s-1}\}$$

$$+ I_x(p+s,q) \ . \qquad (3.7)$$

Let s go to ∞ . Then equation (3.6) converges for $x \leq 1/2$, while (3.7) converges for all x . If q is integer then (3.6) converges for all x .

To examine the rate of convergence for both (3.6) and (3.7), consider the ratio of the density functions. Let y be the density function of $I_x(p,q)$, also let y_1 and y_2 be the density functions of $I_x(p+1,q-1)$ and $I_x(p+1,q)$, respectively. Then solving for the range of x values such that $\frac{y_1}{y_2} < 1$, results in the inequality $x < \frac{p}{p+q}$. With x less than the mean, $I_x(p+1,q-1) < I_x(p+1,q)$. Similarly, it can be shown that $I_x(p+s,q-s) < I_x(p+s,q)$ when $x < \frac{p}{p+q}$ and $q-s$ is positive.

3.4 Continued Fractions for the Incomplete Beta Function

Four continued fractions for the incomplete beta function will be described in this section. For both power series, (3.6) and (3.7), an equivalent continued fraction and a corresponding continued fraction are examined. The difference between an equivalent continued fraction and a corresponding continued fraction can be seen in terms of their definitions. The n^{th} convergent of an equivalent continued fraction is the sum of the first n terms of the related power series, while the n^{th} convergent of a corresponding continued fraction is the sum of the first n terms of the related power series, plus an approximation of the remainder of the related power series. Hence, if the related power series does not terminate, the corresponding continued fraction usually converges faster than the equivalent continued fraction.

Consider equation (3.7). Muller (1930) derives an equivalent continued fraction as follows.

Let $$s = \sum_{k=o}^{\infty} \frac{q(q-1)\ \dots\ (q-k)}{p(p+1)\ \dots\ (p+k)} \left(\frac{x}{1-x}\right)^k$$

and $$s_r = \sum_{k=r}^{\infty} \frac{q(q-1)\ \dots\ (q-k)}{p(p+1)\ \dots\ (p+k)} \left(\frac{x}{1-x}\right)^k .$$

Then
$$s_r - s_{r+1} = \frac{q(q-1)\ \dots\ (q-r)}{p(p+1)\ \dots\ (p+r)} \left(\frac{x}{1-x}\right)^r$$

and
$$s_{r-1} - s_r = \frac{q(q-1)\ \dots\ (q-r+1)}{p(p+1)\ \dots\ (p+r-1)} \left(\frac{x}{1-x}\right)^{r-1} .$$

Therefore,
$$1 - \frac{s_{r+1}}{s_r} = \frac{q-r}{p+r} \left(\frac{x}{1-x}\right)\left(\frac{s_{r-1}}{s_r} - 1\right) ,$$

or $$\frac{s_r}{s_{r-1}} = \frac{\frac{q-r}{p+r}\left(\frac{x}{1-x}\right)}{1 + \frac{q-r}{p+r}\left(\frac{x}{1-x}\right) - \frac{s_{r+1}}{s_r}} .$$

Given that $s_0 = s_1 + \frac{q}{p}$,

$$\frac{s_0}{1+s_0} = \frac{\frac{q}{p}}{1 + \frac{q}{p}} - \frac{\frac{q-1}{p+1}\left(\frac{x}{1-x}\right)}{1 + \frac{q-1}{p+1}\left(\frac{x}{1-x}\right)} - \frac{\frac{q-2}{p+2}\left(\frac{x}{1-x}\right)}{1 + \frac{q-2}{p+2}\left(\frac{x}{1-x}\right)} - \dots \qquad (3.8)$$

If $F = \frac{s_0}{1+s_0}$ then , $s_0 = \frac{F}{1-F}$ hence,

$$I_x(p,q) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q+1)} x^p (1-x)^{q-1} \left(\frac{F}{1-F}\right) .$$

Similarly, for equation (3.7), let

$$s_r = \sum_{k=r}^{\infty} \frac{(p+q-1)(p+q) \ldots (p+q+k-1)}{(p)(p+1) \ldots (p+k)} x^k$$

for $r = 0, 1, \ldots$

By using the procedure used to derive an equivalent fraction for (3.6), the following is obtained:

$$\frac{s_0}{1+s_0} = \frac{\frac{p+q-1}{p}}{1 + \frac{p+q-1}{p}} - \frac{(\frac{p+q}{p+1})\, x}{1 + (\frac{p+q}{p+1})\, x} - \frac{(\frac{p+q+1}{p+2})\, x}{1 + (\frac{p+q+1}{p+2})\, x} - \ldots \tag{3.9}$$

Letting $F = \frac{s_0}{1+s_0}$,

$$I_x(p,q) = \frac{\Gamma(p+q-1)}{\Gamma(p-1)\Gamma(q)} x^p (1-x)^q \frac{F}{1-F} \quad .$$

Muller (1930) makes use of an algorithm, given by Muir (1876), for converting a power series into a corresponding continued fraction. Muller applied Muir's method to (3.6). The resulting corresponding continued fraction has the form:

$$I_x(p,q) = \frac{\Gamma(p+q)}{\Gamma(p+1)\Gamma(q)} x^p (1-x)^{q-1} [\frac{c_1}{1} + \frac{c_2}{1} + \frac{c_3}{1} + \ldots] ,$$

where $c_1 = 1$

$$c_{2k} = \frac{-(p+k-1)(q-k)}{(p+2k-2)(p+2k-1)} (\frac{x}{1-x})$$

$$c_{2k+1} = \frac{k(p+q-1+k)}{(p+2k-1)(p+2k)} \left(\frac{x}{1-x}\right)$$

for $k = 1, 2, \ldots$ (3.10)

If q is integer, the continued fraction terminates after $2k-1$ convergents.

Following Muller's approach, Aroian (1941), applied Muir's method to (3.7). The resulting corresponding continued fraction has the form:

$$I_x(p,q) = \frac{\Gamma(p+q)}{\Gamma(p+1)\Gamma(q)} x^p (1-x)^q \left[\frac{1}{1} + \frac{c_2}{1} + \frac{c_3}{1} + \ldots\right]$$

where $$c_{2k} = -\frac{(p+k-1)(p+q+k-1)}{(p+2k-2)(p+2k-1)} x$$

$$c_{2k+1} = \frac{k(q-k)}{(p+2k-1)(p+2k)} x$$

for $k = 1, 2, \ldots$ (3.11)

3.5 Existing Algorithms

Many methods for calculation of the incomplete beta function have been proposed. One approach is to approximate the beta with another distribution. Peizer and Pratt (1968) use a normal approximation for tail probabilities. The problem with such approximations in general is their limited accuracy over other than a very restricted range of x-values and parameters. Di Donato and Jarnagin (1967) suggest an algorithm which computes the incomplete beta function for the special case of half-integer parameter values. Another approach using three-term recurrence relations is proposed by Gautshi (1964). The problem with this algorithm is the need to evaluate $I_x(p',q)$ accurately for $0 < p' < 1$. Osborn and Madey (1968) use the hypergeometric series to evaluate the incomplete beta function.

Two power series methods are of interest. The MDBETA subroutine from the IMSL library uses two power series. They are

$$s_1 = \sum_{i=1}^{\infty} \frac{p(1-rq)(2-rq)\ \ldots\ (i-rq)}{p+i} \frac{x^i}{i!} ,$$

and

$$s_2 = \sum_{i=1}^{[q]} \frac{q(q-1)\ \ldots\ (q-i+1)}{(p+q-1)(p+q-2)\ \ldots\ (p+q-i)} \frac{1}{(1-x)^i} ,$$

where $[q]$ is the largest integer less than q, $rq = q-[q]$, and $s_2 = 0$ if $q = 1$. $I_x(p,q)$ can be computed as,

$$I_x(p,q) = \frac{x^p(1-x)^{rq}\Gamma(rq+p)}{\Gamma(rq)\Gamma(p+1)} (1+s_1) + \frac{x^p(1-x)^q\Gamma(p+q)}{\Gamma(q)\Gamma(p+1)} s_2 .$$

The series s_1 is only summed until the last term is less than an arbitrarily small number. Equations (3.6) and (3.7) are used by Majumder and Bhattacharjee (1973a) to evaluate the incomplete beta function.

Two algorithms using continued fractions have been proposed. Boardman and Kopitzke (1975) use the continued fraction developed by Muller (1930). The second algorithm is given by Bouver and Bargmann

(1975) and employs the continued fraction due to Aroian (1941) and a Hermite polynomial expansion. The advantages and drawbacks of the last three algorithms described will be discussed in chapter 5.

4. MULTIPLE PRECISION

4.1 Literature

When programs are implemented on different computers, the number of significant digits in the result will vary. This is due to the word-size and mode of numeric representation that the manufacturer has chosen to use. Hence, an algorithm that provides 15 decimal place results for a large range of parameters on one machine, may provide at best only 9 decimal place results for a significantly smaller range of parameters, when implemented on another machine. Also, consider that sometimes scientific results need to be accurate to a large number of digits because they enter into other computations. Both of the above examples are situations in which MP (multiple precision) software might be implemented to provide a guaranteed number of significant digits in a portable program. MP arithmetic is significantly slower than the standard software arithmetic. The slowness is because software must be used to emulate a hardware function.

Most existing MP packages have one or more of the following faults: use of fixed point arithmetic, nonportability, fixed precision, use of ALGOL as the language, or the lack of simple functions such as log or exp. Both Brent (1978), and Wyatt, Lozier, and Orser (1976) provide a list of some existing MP packages in the articles presenting their MP packages. Brent's package allows the user to specify the base (b) and number of digits to carry in the chosen base, with the restriction that $8b^2-1$ be representable in an integer word. Wyatt, Lozier, and

Orser's package allows a base of $b = 2^k$, for $k = 1,2,3$ or 4, and any number of base 2^k digits. While Wyatt, Lozier, and Orser use packed decimal representation to minimize the space required, Brent uses one integer word per digit. Since Brent does not have to pack and unpack numbers, Brent's routines run faster. In computers today, space is not the most important consideration, rather minimizing execution time is of greater concern. Both packages have a common drawback when considered for use in the current applications. The user must have sufficient knowledge of the algorithms used to specify enough digits in the MP subroutines to insure the results have the desired accuracy. Hence, a special set of MP subroutines, that provide desired accuracy, was designed and implemented for the evaluation of the incomplete beta function. These subroutines guarantee that at least the specified accuracy is achieved.

4.2 Description of Multiple Precision Subroutines

The arithmetic operations of MP addition, subtraction, multiplication, and division have well-known algorithms for implementation on a computer. The amount of absolute error incurred in the above arithmetic operations is easily found. However, the absolute error in the evaluation of a simple function, such as the natural logarithm, is unknown unless an algorithm which provides an error bound is used. In the MP subroutines for simple functions used in the IBETA program, desired accuracy is achieved.

To represent a MP number, an integer array is used. MP numbers are stored in a normalized floating point representation using a base of 10,000. The representation used in the IBETA program is the same as Brent (1978) used, with the base fixed as 10,000. To represent a floating point MP number containing N decimal places, an array of $[\frac{N}{4}] + 3$ integer words is needed, where [ ] is the greatest integer less than. The elements of an array, in which a MP number is stored, have the following functions. The first element contains an indicator of the sign of the MP number, with +1 denoting a positive number, -1 representing a negative number, and 0 implying the number is zero. To represent the exponent, the second element of the array is used. Note this element is a signed integer, indicating whether the exponent is negative or positive. The elements 3 through $[\frac{N}{4}] + 3$ are the mantissa in normalized base 10,000 digits. The precision of a MP number, that is stored in an array of size $[\frac{N}{4}] + 3$, is at least N decimal digits. The maximum precision, that can be achieved, is N+4 decimal digits. The reason for the changing precision is due to wobbling of the third element of the array between the values of 1 and 9999. It should be noted that in the representation of zero that all elements of the array, other than the first element are undefined. In the remainder of this chapter, reference to a digit will mean a base 10,000 digit, unless it is specifically referenced as a decimal digit.

4.2.1 Addition and subtraction

Addition and subtraction are performed with two guard digits. The smaller of the two operands in absolute value is shifted so that the exponents of both operands are equal. The operation of addition or subtraction is then performed using algorithms given by Knuth (1981) The result is returned in normalized truncated form.

4.2.2 Multiplication

Knuth (1981) describes the classical algorithm for multipling a n-digit integer by a m-digit integer with the result being a $m+n$-digit integer. Hence, if MP numbers have t digits, the result of a multiplication will be an unnormalized MP number containing $2t$ digits. The result will then be normalized and truncated to t digits. If $M(n)$ denotes the time needed to perform n integer multiplications, then the time need to multiple t-digit MP numbers is of the order $M(t^2)$. It is possible to multiply 2n-bit binary numbers in $O(n^{\log_2(3)})$ time. The algorithm is given in Knuth (1981), but the method is not applicable to base 10,000 digits.

4.2.3 Division

To perform MP division, the reciprocal of the divisor is found and then multiplied by the numerator. Reciprocation is performed using a third order Newton's method described in Knuth (1981). The algorithm is iterative, with the number of correct digits being tripled after each iteration. To minimize execution time, the number of digits carried on each successive iteration is tripled. The number of digits carried on the first iteration is two. To find $\frac{1}{v}$ the iteration equation

$$x_{n+1} = x_n(1 + (1-vx_n)(1 + (1-vx_n)))$$

is used. Hence, each iteration requires 3 MP multiplications.

4.2.4 Integer MP division and multiplication

Since MP division and multiplication require $O(M(t^2))$ time, subroutines for multiplication and division of a MP number by an integer have been implemented. The time needed to perform an integer multiplication or division of a MP number is $O(M(t))$. Therefore, the use of integer multiplication or division when possible will greatly reduce execution time.

4.2.5 Exponential function

To evaluate $\exp(x)$ an algorithm given by Cody and Waite (1980) is implemented. The idea is to rewrite the value x in a form that facilitates the computation of the exponential function.

If

$$x = N * \ln(10) + g \qquad (4.1)$$

where N is integer and $|g| \le \frac{\ln(10)}{2}$, then

$$\exp(x) = \exp(g) \cdot 10^N . \qquad (4.2)$$

Letting [ ] be the greatest integer less than, the above expression can be rewritten to take advantage of the chosen MP base of 10,000. Hence,

$$\exp(x) = \exp(g) * 10{,}000^{[\frac{N}{4}]} * 10^{N-[\frac{N}{4}]} . \qquad (4.3)$$

After exp(g) is evaluated, at most only 1 more integer MP multiplication is needed to find exp(x) . To further reduce the argument g , the relationship

$$\exp(g) = [\exp(\frac{g}{2^k})]^{2^k}$$

is used. This relationship is applied until $|\frac{g}{2^k}| \leq .0008$. If $|g| > .0008$ then equation (4.3) becomes

$$\exp(x) = \exp(\frac{g}{2^k})^{2^k} * 10{,}000^{[\frac{N}{4}]} * 10^{N-[\frac{N}{4}]} \quad . \tag{4.4}$$

To evaluate exp(g) or $\exp(\frac{g}{2^k})$ a continued fraction is used. The form of the continued fraction as given by Khovanskii (1956) is

$$e^x = 1 + \frac{x}{1} - \frac{x}{2} + \frac{x}{3} - \frac{x}{2} + \frac{x}{5} \ldots - \frac{x}{2} + \frac{x}{2n+1} \ldots \tag{4.5}$$

Peizer and Pratt (1968) have shown that for $|x| < 2$ the convergents 2, 3, 6, 7, 10, 11, ... form a monotonic decreasing sequence above the value e^x , while the convergents 4, 5, 8, 9, 12, 13, ... form a monotonic increasing sequence below the value of e^x . Hence, the convergents of the continued fraction provide an error bound for terminating the continued fraction at the nth convergent.

To evaluate exp(x) the values ln(10) and $\frac{1}{\ln(10)}$ have been precomputed to 112 decimal digits. Equation (4.1) is solved for N and g using the precomputed values of $\ln(10)$ and $\frac{1}{\ln(10)}$. The argument g is then reduced using integer MP division until $\frac{g}{2^k} < .0008$. The computation of the continued fraction (4.5) is by the forward method based on equations (2.1) and (2.2). Since the partial denominators are integers, 2 MP multiplications, 1 MP division, 2 MP additions and 2 integer MP multiplications are needed to evaluate a convergent. The value of $\exp(x)$ is then reconstructed using either equation (4.3) or (4.4), as is appropriate. To guard against round off error, 3 extra guard digits are used in the computation of $\exp(x)$.

This method of evaluating $\exp(x)$ has an advantage over evaluating the power series expansion of $\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}$. In the forward evaluation of the power series, to assure n digits of accuracy at least $2n$ digits should be carried in the summation. Also, the absolute error after summing the first r terms is unknown, and must be approximated to insure that the desired accuracy is achieved. However, by using the continued fraction (4.5), error bounds are generated in the computation of $\exp(x)$.

4.2.6 Natural logarithm function

Consider the following representation of an MP number x.

$$x = f * 10{,}000^m ,$$

where f is a fraction less than 1. Hence,

$$\ln(x) = 4 * m * \ln(10) + \ln(f) . \qquad (4.6)$$

The algorithm used to compute ln(x) is a modification of the ALOG10 algorithm by Cody and Waite (1980). The value f is rewritten as

$$f = f' * 10^{-n}$$

where $1 \leq f' < 10$. Then the relationship

$$\ln(x) = \ln(\frac{x}{2^k}) + k * \ln(2)$$

is then used to reduce the value for which the ln must be computed, to a value greater than or equal to one but less than two. Let k be the largest integer such that $f'/2^k$ is greater than or equal to 1 . The value k assumes is between 0 and 3 . If $r = f'/2^k$ equation (4.6) becomes

$$\ln(x) = (4*m-n) * \ln(10) + \ln(r) + k * \ln(2) . \qquad (4.7)$$

To compute the value of ln(x) the values ln(10) and ln(2) have been precomputed to 112 decimal places. A continued fraction given by Khovanskii (1956) is used to evaluate ln(r) . The general form of the continued fraction is

$$\ln(r) = \frac{z}{1} + \frac{z}{2} + \frac{z}{3} + \frac{2z}{2} + \frac{2z}{5}$$

$$+ \ldots \frac{nz}{2} + \frac{nz}{2n+1} + \frac{(n+1)z}{2} + \frac{(n+1)z}{2n+3} + \ldots , \qquad (4.8)$$

where $r-1 = z$. Since $1 \leq r < 2$, $0 \leq z < 1$ and the even convergents of equation (4.8) form an increasing sequence of lower bounds for the value ln(r) . Likewise, the odd convergents form a decreasing sequence of upper bounds for ln(r) . This property of convergence of the continued fraction is given in Chapter 2 and applies to equation (4.8) since all partial numerators and denominators are positive. The convergents of the continued fraction (4.8) are evaluated until an acceptable accuracy is achieved. As in the evaluation of exp(x) , 2 MP multiplications, 1 MP division, 2 integer MP multiplications, and 2 MP additions are required to evaluate each convergent. The same advantages of using a continued fraction to evaluate exp(x) , instead of evaluating the power series expansion of exp(x) , apply to the evaluation of ln(x) .

4.2.7 ln(Γ(x))

To compute the ln(Γ(x)) function, Stirling's approximation is evaluated. Stirling's approximation has the form

$$\ln(\Gamma(x)) = (x - \frac{1}{2}) \ln(x) - x + \frac{1}{2} \ln(2\pi) + \sum_{k=1}^{\infty} \frac{B_{2n}}{(2n-1)(2n)x^{2n-1}} , \qquad (4.9)$$

where B_{2n} are the Bernoulli numbers as defined in Abramowitz and Stegun (1964). The error in approximating ln(Γ(x)) with a finite number of terms of the infinite sum is less than the absolute value of the first term not computed.

In order to keep from computing an arbitrary number of Bernoulli numbers the integer j is found such that $\frac{B_{60}}{(60)(59)(x+j)^{59}}$ is sufficiently small. The first 29 Bernoulli numbers are stored in a form such that they can be calculated by using at most 2 integer MP division per number. If j is zero a double precision search is made for the smallest number of terms in the series in equation (4.9) that need be computed to achieve sufficient accuracy. When j is not zero, the equation

$$\Gamma(x+j) = \prod_{i=0}^{j-1} (x+i)\Gamma(x) \tag{4.10}$$

is used to find $\ln(\Gamma(x))$ from $\ln(\Gamma(x+j))$. The value of $\ln(\Gamma(x+j))$ is computed and equation (4.10) applied with the result that

$$\ln(\Gamma(x)) = - \ln(\prod_{i=0}^{j-1} (x+j)) + \ln(\Gamma(x+j)) .$$

The necessary number of terms are computed and summed from smallest to largest to avoid excessive round-off error.

5. COMPUTATION OF THE INCOMPLETE BETA FUNCTION BY THE SUBROUTINE IBETA

5.1 Objectives

The purpose of this research is to design and implement an algorithm for the evaluation of the incomplete beta function to a desired number of significant decimal digits. In this chapter, reference to a digit will imply a decimal digit. The program developed must be portable with the exception of the subroutines for conversion between machine representable numbers and MP numbers. To decrease execution time, the maximum number of significant digits a user can ask for is limited to 70. The limitation on the maximum number of significant digits allows precomputation of several needed constants.

Three versions of the program IBETA, which evaluates $I_x(p,q)$, have been programmed. The different versions allow the user to specify what type of precision the arguments assume in the subroutine IBETA. The first version of IBETA is accessed by the fortran call statement:

```
CALL IBETA1 (P, Q, X, N, ANS).
```

The arguments P, Q, and X are double precision numbers and correspond to the parameters of $I_x(p,q)$. N is an integer variable specifying the desired number of significant decimal digits in the value $I_x(p,q)$. The value $I_x(p,q)$ is computed in multiple precision. The multiple precision representation of $I_x(p,q)$ is then converted to double precision and returned in the double precision variable ANS .

The call statement of the second version of IBETA has the form

```
CALL IBETA2 (P, Q, X, N, ANS) .
```

The arguments P, Q, and X are as defined in the subroutine IBETA1, and are double precision numbers. N is as defined in IBETA1. The value of $I_x(p,q)$, in this version of IBETA, is returned as a MP number in the integer array ANS . The array ANS must be dimensioned with size S , where $S = [\frac{N}{4}] + 3$. The definitions of the elements of ANS are as given in Chapter 4, Section 2.

The third version of IBETA allows the user to specify multiple precision parameters. The form of the call statement is:

```
CALL IBETA3 (P, Q, X, S, ANS).
```

The arguments P, Q, X, and ANS are integer arrays of size S . The user must represent the parameters of $I_x(p,q)$ in the MP form given in Chapter 4, Section 2.

5.2 Bounding Error

To guarantee a correct number of significant digits, two sources of error must be controlled. The first source of error involves the evaluation of $\frac{\Gamma(p+q)}{\Gamma(p+1)\Gamma(q)} x^p(1-x)^q$ or a similar expression. The other source of error comes from evaluating or approximating a power series related to the above expression, (see equations (3.6) and (3.7)). If both of the above errors are limited to an acceptable level, then the needed number of correct significant digits can be obtained.

Consider the computation of $\frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} x^p(1-x)^q$. To avoid exponent underflow due to exponentiation of x or $1-x$, when p or q are large or exponent overflow in computing $\Gamma(p+q)$, the quantity

$$\ln(\Gamma(p+q)) - \ln(\Gamma(p+1)) - \ln(\Gamma(q)) + p\ln(x) + q\ln(1-x) \qquad (5.1)$$

is evaluated. It is assumed that p, q, and x are given to the program exactly. The algorithms for $\ln(\Gamma(x))$ and $\ln(x)$ given in Chapter 4, will return values correct to a specified number of extended digits when the argument x is specified with no error. To transform equation (5.1) back to $\frac{\Gamma(p+q)}{\Gamma(p+1)\Gamma(q)} x^p(1-x)^q$ the exponential function is used. Consider the algorithm used to evaluate $\exp(x)$ given in Chapter 4.

From equations (4.3) and (4.4), it can be seen that the relative error in the computation of $\exp(x)$ is approximately the absolute error in the evaluation of $\exp(g)$ or $\exp(g/2^k)$. Since $\exp(g)$ lies in the interval $(.3, 3.2)$ the relative error in the evaluation of $\exp(x)$ is approximately the absolute error in $\exp(g)$. To achieve small relative error in $\exp(g)$, the absolute error in the computation of g from equation (4.1) must be acceptable (see Cody and Waite (1980), page 60). The magnitude of the value N as defined in equation (4.1) determines the absolute error in g . The largest value N can obtain before exponent overflow or underflow will occur, is approximately 2×10^9 . Hence, if a result with t extended significant digits is desired, t+3 extended digits must be used in the evaluation of g or t+3 digits must be used to evaluate (5.1).

To evaluate the power series associated with equation (5.1), forward summation of the power series is not considered. Since to achieve a result accurate to t extended digit in the forward summation, $2t$ extended digits should be carried in the computations. Hence, only continued fractions are considered for the evaluation of equations (3.6) and (3.7). One necessary property of any continued fraction under consideration is the bounding of the truncation error. If the n^{th} convergent is used as an approximation of the related power series, then either the error must be explicitly known or upper and lower bounds on the error must be computable. Under the stipulation that truncation error is computable, Aroian's continued fraction (3.11) is not acceptable since the truncation error is unknown. The convergents of Muller's continued fraction (3.10) have been shown by Peizer and Pratt (1968), to form bounds on truncation error, for certain ranges of p, q, and x. Hence, Muller's continued fraction is an acceptable method of evaluating the incomplete beta function for some combinations of p, q, and x. Soper (1921) gives truncation bounds for the power series (3.6) and (3.7), and hence, those bounds are applicable to the related equivalent continued fractions.

5.3 Approximation Error in Continued Fractions for the Incomplete Beta Function

Consider Muller's continued fraction for the incomplete beta function.

$$I_x(p,q) = \frac{\Gamma(p+q)}{\Gamma(p+1)\Gamma(q)} x^p (1-x)^{q-1} [\frac{c_1}{1} + \frac{c_2}{1} + \ldots] ,$$

where

$$c_1 = 1$$

$$c_{2k} = - \frac{(p+k-1)(q-k)}{(p+2k-2)(p+2k-1)} (\frac{x}{1-x})$$

$$c_{2k+1} = \frac{k(p+q-1+k)}{(p+2k-1)(p+2k)} (\frac{x}{1-x})$$

for $k = 1, 2, 3, \ldots$ (5.2)

The behavior of the convergents of equation (5.2), are determined by the partial numerators c_{2k} . If $q < 1$ then all partial numerators are positive. From the results given in Chapter 2, the even convergents form an increasing sequence of lower bounds for the continued fraction, while the odd convergents form a decreasing sequence of upper bounds for the continued fraction. If $q > 1$, consider c_{2k} for $k = 1, 2, \ldots, [q]$, where [ ] is the greatest integer less than. The partial numerator

$$c_{2k} = - \frac{(p+k-1)(q-k)}{(p+2k-2)(p+2k-1)} (\frac{x}{1-x})$$

is an increasing function of k . Hence, the largest term is absolute value is c_2 . Using repeated applications of the fact that $\frac{1}{1+x}$ is a decreasing function of x for $x > -1$, it can be shown that the convergents provide error bounds for using the n^{th} convergent to approximate the continued fraction when $c_2 > -1$. Solving for x such that

$$-\frac{(q-1)}{(p+1)}\left(\frac{x}{1-x}\right) > -1 \text{ ,}$$

results in the restriction that

$$x < \frac{p+1}{p+q} \text{ .}$$

Hence, for $x < \frac{p+1}{p+q}$ the convergents $n = 2, 3, 6, 7, \ldots$ form decreasing upper bounds, while the convergents $n = 4, 5, 8, 9, \ldots$ form increasing lower bounds when $n \le 2[q]-1$. For $n > 2[q]-1$, the n^{th} and $n{+}1^{th}$ convergents bound the value of the continued fraction.

The truncation error incurred, if the n^{th} convergent of the continued fraction (3.8) is used to approximate the continued fraction, can be found by examining the related power series. Consider,

$$I_x(p,q) = \frac{\Gamma(p+q)}{\Gamma(p-1)\Gamma(q+1)}\, x^p (1-x)^{q-1} \frac{F}{1-F} \text{ ,} \tag{5.3}$$

where $$F = \frac{s_0}{1+s_0} = \frac{c_1}{1+c_1} - \frac{c_2}{1+c_2} - \frac{c_3}{1+c_3} - \ldots$$

and $$c_1 = \frac{q}{p}$$

$$c_n = \frac{q-n+1}{p+n-1}\left(\frac{x}{1-x}\right)$$

for $n = 2, 3, \ldots$.

The equation (5.3) is an equivalent continued fraction for the related power series (3.6). By the definition of an equivalent continued fraction, the value of the nth convergent is the sum of the first n terms of the related power series. Therefore, using equation (3.6) the truncation error for the nth convergent is $I_x(p+n, q-n)$. Let $p' = p+n$ and $q' = q-n$. Then the power series expansion of $I_x(p',q')$ is

$$I_x(p',q') = \frac{\Gamma(p'+q')}{\Gamma(p'+1)\Gamma(q')} x^{p'} (1-x)^{q'-1} [1 + \frac{(q'-1)}{(p'+1)} (\frac{x}{1-x}) + \frac{(q'-1)(q'-2)}{(p'+1)(p'+2)} (\frac{x}{1-x})^2 + \frac{(q'-1)(q'-2)(q'-3)}{(p'+1)(p'+2)(p'+3)} (\frac{x}{1-x})^3 + \ldots] \tag{5.4}$$

Let

$$r = \frac{(q'-1)}{(p'+1)} (\frac{x}{1-x}) .$$

Assume $\frac{(q'-1)}{(p'+1)} (\frac{x}{1-x}) < 1$ or equivalently $x < \frac{p'}{p'+q'}$ (i.e., the mean has been driven past x). Then, by using the summation formula for a geometric series,

$$I_x(p',q') < \frac{\Gamma(p'+q')}{\Gamma(p'+1)\Gamma(q')} x^{p'} (1-x)^{q'-1} (\frac{1}{1-r}) . \tag{5.5}$$

Now consider another power series expansion of $I_x(p',q')$ based on equations (3.6) and (3.7).

$$I_x(p',q') = \frac{\Gamma(p'+q')}{\Gamma(p'+1)\Gamma(q')} x^{p'} (1-x)^{q'-1} [1 + \frac{q'-1}{p'+1} x$$

$$+ \frac{(q'-1)}{(p'+1)} \frac{(p'+q')}{(p'+2)} x^2$$

$$+ \frac{(q'-1)(p'+q')(p'+q'+1)}{(p'+1)(p'+2)(p'+3)} x^3 + \dots] \tag{5.6}$$

If in equation (5.6) $q'-1 > 0$, then combined with inequality (5.5)

$$\frac{\Gamma(p'+q')}{\Gamma(p'+1)\Gamma(q')} x^{p'} (1-x)^{q'-1} < I_x(p',q') <$$

$$\frac{\Gamma(p'+q')}{\Gamma(p'+1)\Gamma(q')} x^{p'} (1-x)^{q'-1} (\frac{1}{1-r}) \quad . \tag{5.7}$$

It should be remembered that for $x > 1/2$ the power series (3.6) diverges.

The last continued fraction to be examined is based on equation (3.7).

$$I_x(p,q) = \frac{\Gamma(p+q-1)}{\Gamma(p-1)\Gamma(q)} x^p (1-x)^q \frac{F}{1-F} \tag{5.8}$$

where

$$F = \frac{s_0}{1+s_0} = \frac{c_1}{1+c_1} - \frac{c_2}{1+c_2} - \frac{c_3}{1+c_3} - \dots$$

with

$$c_1 = \frac{p+q-1}{p-1} \quad ,$$

and

$$c_{n+1} = \frac{p+q-1+n}{p+n-1}\, x$$

for $n = 1, 2, \ldots$.

Again, by construction this continued fraction is an equivalent continued fraction. The error in terminating at the n^{th} convergent is $I_x(p+n,q)$. Letting $p' = p+n$, the power series expansion of $I_x(p',q)$ is

$$I_x(p',q) = \frac{\Gamma(p'+q)}{\Gamma(p'+1)\Gamma(q)}\, x^{p'} (1-x)^q \, [1 + \frac{p'+q}{p'}\, x + \frac{(p'+q)(p'+q+1)}{p'(p'+1)}\, x^2 + \frac{(p'+q)(p'+q+1)(p'+q+2)}{p'(p'+1)(p'+2)}\, x^3 + \ldots] \tag{5.9}$$

If $x < \frac{p'}{p'+q}$ then using equation (5.9) ,

$$\frac{\Gamma(p'+q)}{\Gamma(p'+1)\Gamma(q)}\, x^{p'} (1-x)^q \, (\frac{1}{1-x}) < I_x(p',q) < \frac{\Gamma(p'+q)}{\Gamma(p'+1)\Gamma(q)}\, x^{p'}(1-x)^q \frac{1}{1 - (\frac{p'+q}{p'})x} \; . \tag{5.10}$$

The continued fraction (5.8) converges for all $x < 1$.

5.4 Method of Computation of the Incomplete Beta Function

The choice of which of the three continued fractions (5.3), (5.4), or (5.8) to use to evaluate the incomplete beta function depends on the values of p, q, and x. For $x > 1/2$ the continued fractions (5.3) and (5.4) will diverge. However, for $x > 1/2$, the continued fraction (5.4) can be evaluated to the $[q]^{th}$ convergent and will generally have smaller truncation error than the continued fraction (5.8) evaluated to the $[q]^{th}$ convergent. At this point, switching to the continued fraction (5.8) for the evaluation of the remainder $I_x(p+[q], q-[q])$ is a method for evaluating the incomplete beta function.

Soper (1921), has shown that if $x > \frac{p}{p+q}$ then the evaluation of $I_{1-x}(q,p)$ will converge faster than the evaluation of $I_x(p,q)$. Equation (3.1) gives the relationship between $I_{1-x}(q,p)$ and $I_x(p,q)$. Hence, if $I_{1-x}(q,p)$ has been calculated then $I_x(p,q) = 1-I_{1-x}(q,p)$. Soper (1921), also shows that the first $[q]$ convergents of the continued fraction (5.4) converge faster than the first $[q]$ convergents of continued fraction (5.8) if $x < \frac{p}{p+q}$. Therefore, it is advisable to use continued fraction (5.8) only when continued fraction (5.4) does not provide sufficient accuracy after $[q]$ convergents, since using equation (3.1) there is no need to evaluate $I_x(p,q)$ for $x > \frac{p}{p+q}$.

Of the three continued fractions, Muller's corresponding continued fraction (5.3) generally converges fastest (see Section 3.4). Therefore, if $x < 1/2$ and $x < \frac{p}{p+q}$, Muller's continued fraction is used to evaluate the incomplete beta function $I_x(p,q)$. If $x < 1/2$ but

$x > \frac{p}{p+q}$, equation (3.1) is applied and $I_{1-x}(q,p)$ is evaluated by computing continued fraction (5.4) for up to [p] convergents. If sufficient accuracy has not been achieved, the continued fraction (5.8) is employed to evaluate $I_{1-x}(q+[p], p-[p])$ to desired accuracy. The value of $I_x(p,q)$ is then computed using the values found using continued fractions (5.4) and (5.8). If $x > 1/2$ and $x < \frac{p}{p+q}$ then the continued fractions (5.4) and (5.8) are used to compute $I_x(p,q)$. The last case to be considered is $x > 1/2$ and $x > \frac{p}{p+q}$ in this case $1-x < 1/2$ and $1-x < \frac{q}{p+q}$. Hence, Muller's continued fraction is used to evaluate $I_{1-x}(q,p)$ and $1-I_{1-x}(q,p)$ is returned as the value of $I_x(p,q)$. The method using continued fractions (5.4) and (5.8), is implemented as the subroutine RPLQ in the program IBETA given in Appendix A. Muller's continued fraction is implemented in the subroutine MULLER in the program given in Appendix A.

5.5 Computational Aspects of the Subroutine MULLER

In performing multiple precision (MP) arithmetic, MP multiplications are extremely time consuming. Hence, if MP multiplications can be replaced with MP additions and subtractions, the execution time of a MP subroutine can be greatly reduced. This section describes how Muller's continued fraction is computed and the methods employed to minimize execution time.

Consider Muller's continued fraction

$$I_x(p,q) = \frac{\Gamma(p+q)}{\Gamma(p+1)\Gamma(q)} x^p (1-x)^{q-1} \left[\frac{c_1}{1} + \frac{c_2}{1} + \frac{c_3}{1} + \dots \right] ,$$

where

$$c_1 = 1$$

$$c_{2k} = - \frac{(p+k-1)(q-k)}{(p+2k-2)(p+2k-1)} \left(\frac{x}{1-x}\right)$$

$$c_{2k+1} = \frac{k(p+q-1+k)}{(p+2k-1)(p+2k)} \left(\frac{x}{1-x}\right) .$$

The quantity $\frac{\Gamma(p+q)}{\Gamma(p+1)\Gamma(q)} x^p (1-x)^{q-1}$ is computed as described in Section 5.2. To limit the truncation error in using the n^{th} convergent to approximate the continued fraction $\frac{c_1}{1} + \frac{c_2}{1} + \frac{c_3}{1} + \dots$, the convergents are calculated until they bound the continued fraction with sufficient accuracy (see Section 5.3). The method employed to evaluate the convergents w_n is as follows:

$$w_n = \frac{P_n}{Q_n} = \frac{P_{n-1} + c_n P_{n-2}}{Q_{n-1} + c_n Q_{n-2}}$$

with

$$P_{-1} = 1 , P_0 = 0 , Q_{-1} = 0 , Q_0 = 1 ,$$

and c_n are the partial numerators of Muller's continued fraction.

The usual method of calculating the c_n's would require 6 MP multiplications and 1 MP division. The number of MP multiplications and division needed to evaluate c_n can be reduced to 1 MP division, if several quantities are carried along at each calculation of the next convergent. Consider the relationship between the divisors of the partial numerators c_{2k} and c_{2k+1}.

$$(p+2k-2)(p+2k-1)(1-x) = (p^2+4pk-3p+4k^2-6k+2)(1-x)$$

$$(p+2k-1)(p+2k)(1-x) = (p^2+4pk-p+4k^2-2k)(1-x)$$

so

$$(p+2k-2)(p+2k-1)(1-x) + (2p+4k-2)(1-x) = (p+2k-1)(p+2k) \ . \quad (5.11)$$

Now, consider the divisors of the partial numerators c_{2k+1} and $c_{2(k+1)}$.

$$(p+2k-1)(p+2k)(1-x) = (p^2+4pk-p+4k^2-2k)(1-x)$$

$$(p+2(k+1)-2)(p+2(k+1)-1)(1-x) = (p^2+4pk+p+4k^2+2k)(1-x)$$

so

$$(p+2k-1)(p+2k)(1-x) + (2p+4k)(1-x) =$$

$$(p+2(k+1)-2)(p+2(k+1)-1)(1-x) \ . \quad (5.12)$$

Equations (5.11) and (5.12), allow the divisor of c_n to be calculated iteratively without using MP multiplication, if the quantities $2p+4k$ or $2p+4k-2$, $(-2px+4kx-2x)$ or $(-2px+4kx)$, 2, $-2x$, and the divisor of the previous c_n are stored and updated appropriately.

The next 3 MP multiplications to be replaced are in the calculation of the numerator of c_{2k}. Consider the numerators of c_{2k} and $c_{2(k+1)}$.

$$(p+k-1)(q-k)(x) = (pq-k^2-pk+qk+k-q)(x)$$

$$(p+(k+1)-1)(q-(k+1))(x) = (pq-k^2-pk+qk-p-k)(x)$$

so

$$(p+k-1)(q-k)(x) - (p-q+2k)(x) = (p+(k+1)-1)(q-(k+1))(x) \ . \quad (5.13)$$

The numerators of the c_{2n}'s can be calculated iteratively if the numerator of the previous even partial numerator has been stored and the quantity $px-qx+2kx$ has been stored and updated appropriately.

Similarly, consider the numerators of c_{2k+1} and c_{2k+3} .

$$k(p+q-1+k)(x) = (k^2+kp+kq-k)(x)$$

$$(k+1)(p+q-1+k+1)(x) = (k^2+kp+kq+p+q+k)(x)$$

Hence,

$$k(p+q-1+k)(x) + px+qx+2kx = (k+1)(p+q-1+k+1)(x) \ . \quad (5.14)$$

As before with storage of appropriate values, no MP multiplications are needed to evaluate the numerator of c_{2k+1} . Using equations (5.11), (5.12), (5.13), and (5.14) each c_n can be evaluated with only one MP division. The time to evaluate each convergent has been reduced by the replacement of 6 MP multiplications.

5.6 Computational Aspects of the Subroutine RPLQ

The subroutine RPLQ computes the incomplete beta function by evaluating the continued fraction (5.3) for $I_x(p,q)$ and if needed the continued fraction (5.8) for $I_x(p+[q], q-[q])$. Consider the partial numerators of the continued fraction (5.3).

$$c_1 = \frac{q}{p}$$

$$c_n = \frac{q-n+1}{p+n-1}$$

The error if the first convergent is used to approximate the continued fraction can be expressed as follows:

$$\frac{\Gamma(p+q)}{\Gamma(p+2)\Gamma(q-1)} x^{p+1}(1-x)^{q-2} < I_x(p+1,q-1) <$$

$$\frac{\Gamma(p+q)}{\Gamma(p+2)\Gamma(q-1)} x^{p+1}(1-x)^{q-2} \left(\frac{1}{1 - \frac{q-2}{p+2}\left(\frac{x}{1-x}\right)}\right) . \qquad (5.15)$$

To compute $I_x(p,q)$ using the continued fraction (5.4) the quantity $\frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q+1)} x^p(1-x)^{q-1}$ must be computed. Suppose that $\frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q+1)} x^p(1-x)^{q-1}$ has been stored. Then to calculate the lower bound on the truncation error for the first convergent, the following relation can be used:

$$\frac{\Gamma(p+q)}{\Gamma(p+2)\Gamma(q-1)} x^{p+1}(1-x)^{q-2} = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q+1)} x^p(1-x)^{q-1} c_1 c_2 .$$

Also, the value $\frac{q-2}{p+2}\left(\frac{x}{1-x}\right)$ in equation (5.15) is the partial numerator c_3 .

Now consider that the second convergent is used as the approximation. Also, assume the values $\frac{\Gamma(p+q)}{\Gamma(p+2)\Gamma(q-1)} x^{p+1}(1-x)^{q-2}$, c_2 and c_3 have been computed and stored. In computing the second convergent, the necessary value c_2 has already been found. The truncation error is found as

$$\frac{\Gamma(p+q)}{\Gamma(p+2)\Gamma(q-1)} x^{p+1}(1-x)^{q-2} c_3 < I_x(p+2, q-2) <$$

$$\frac{(p+q)}{(p+2)\ (q-1)} x^{p+1}(1-x)^{q-2} c_3 \frac{1}{(1 - \frac{(q-3)}{(p+3)} (\frac{x}{1-x}))} . \qquad (5.16)$$

It should be noted that $\frac{q-3}{p+3} (\frac{x}{1-x})$ is c_4 . In general, let

$$K_1 = \frac{\Gamma(p+q)}{\Gamma(p+2)\Gamma(q-1)} x^{p+1}(1-x)^{q-2}$$

$$K_2 = K_1 \cdot c_3$$

$$E_1 = I_x(p+1, q-1)$$

$$K_n = K_{n-1} \cdot c_{n+1}$$

$$E_n = I_x(p+n, q-n) \quad \text{for} \quad n = 2, 3, \ldots, [q]-2$$

The value E_n is the truncation error if the n^{th} convergent is used to approximate the continued fraction and the bounds on E_n have the form: $K_n < E_n < K_n (\frac{1}{1-c_{n+2}})$. Hence, if K_{n-1}, c_n, and c_{n+1} have been kept only the value c_{n+2} need be computed to find the n^{th}

convergent approximation and its associated error bounds.

Consider the convergent $q-[q]+1$. At this point in the algorithm one more convergent is evaluated and the truncation error is $I_x(p+[q], q-[q])$. The subroutine RPLQ now uses the continued fraction (5.8) to evaluate $I_x(p+[q], q-[q])$. The value of $\frac{\Gamma(p+q-1)}{\Gamma(p+[q]-1)\Gamma(q-[q])}$ $x^{p+[q]}(1-x)^{[q]}$ is needed to evaluate equation (5.8). At this point, $K_{[q]-1}$ has the value $\frac{\Gamma(p+q)}{\Gamma(p+[q])\Gamma(q-[q])}x^{p+[q]-1}(1-x)^{q-[q]}$. Hence, $\frac{\Gamma(p+q-1)}{\Gamma(p+[q]-1)\Gamma(q-[q])}x^{p+[q]}(1-x)^{q-[q]} = K_{[q]-1}(\frac{p+[q]}{p+q-1})\, x$. From equation (5.10), the truncation error incurred by not switching to the evaluation of $I_x(p+[q], q-[q])$ is $\frac{\Gamma(p+q)}{\Gamma(p+[q]+1)\Gamma(q-[q])}x^{p+[q]}(1-x)^{q-[q]}$

$$(\frac{x}{1-x}) < I_x(p+[q], q-[q]) < \frac{\Gamma(p+q)}{\Gamma(p+[q]+1)\Gamma(q-[q])}\, x^{p+[q]}(1-x)^{q-[q]}$$

$$\frac{1}{(1-\frac{p+q}{p+[q]}x)} \quad \text{or} \quad K_{[q]}(\frac{x}{1-x}) < I_x(p+[q], q-[q]) < K_{[q]}(\frac{1}{1-(\frac{p+q}{p+[q]})\,x}) \; .$$

In the evaluation of the remainder when the continued fraction (5.8) is used, consider the partial numerators

$$c_1 = \frac{p+q-1}{p+[q]-1}$$

$$c_{n+1} = \frac{p+q+n-1}{p+[q]+n-1}\, x \; .$$

Define,

$$K_1 = \frac{\Gamma(p+q)}{\Gamma(p+[q]+1)\Gamma(q-[q])} x^{p+[q]}(1-x)^{q-[q]}$$

$$K_n = K_{n-1} \cdot c_{n+1}$$

$$E_n = I(p+[q]+n, q-[q])$$

for $n = 1, 2, 3, \ldots$

Then using the inequality (5.10), the truncation error E_n , if the nth convergent is used to approximate $I_x(p+[q], q-[q])$, is bounded as follows:

$$K_n \left(\frac{1}{1-x}\right) \le E_n \le K_n \frac{1}{1-c_{n+2}} .$$

Hence, as with the continued fraction (5.4), if K_n, c_n, c_{n+1} are kept and c_{n+2} is calculated the nth convergent and its truncation error are easily calculated.

Consider the computation of the nth convergent of the continued fractions (5.4) and (5.8). Define F_n as the nth convergent then

$$F_n = \frac{c_1}{1+c_1} - \frac{c_2}{1+c_2} - \frac{c_3}{1+c_3} - \cdots - \frac{c_n}{1+c_n} .$$

Another method of computing F_n is by using equations (2.1) and (2.2).

$$P_1 = c_1, \quad P_2 = c_1(1+c_2), \quad Q_1 = 1+c_1,$$

$$Q_2 = (1+c_2)(1+c_1) - c_2 = 1+P_2$$

and in general, $Q_n = 1+P_n$. Hence,

$$\frac{F_n}{1-F_n} = \frac{\frac{P_n}{1+P_n}}{1 - \frac{P_n}{1+P_n}} = P_n$$

So to evaluate the n^{th} convergent as an approximation to the incomplete beta function, it suffices to compute P_n multiplied by the appropriate constant. The computation of P_n can be simplified by noting that

$$P_n = (1+c_n)P_{n-1} - c_n P_{n-2} = P_{n-1} + c_n(P_{n-1} - P_{n-2}) \ .$$

5.7 Roundoff and Representation Considerations

In Chapter 4 it is shown that to represent a N decimal digit number $[\frac{N}{4}]+1$ base 10,000 digits are needed. Hence, the user should specify the decimal precision N , wished for in the evaluation of the incomplete beta function, so that the values p, q, x, and 1-x are exactly representable in $[\frac{N}{4}]+1$ base 10,000 digits. Otherwise, the result returned will be for a perturbation of the parameters p, q, and x .

To control roundoff error, the program IBETA determines the number of base 10,000 digits (say, N1) necessary to represent the maximum of p^2, q^2, and p+q . Next, the number of extended digits (say, N2 in base 10,000) needed to accurately represent the maximum of p+(1-x), qx-x, and (p+q)x+x is determined. If N1 and N2 are less than the user specified value of $[\frac{N}{4}]+1$, the program IBETA uses $[\frac{N}{4}]+4$ extended digits to solve for the value $I_x(p,q)$. Otherwise, if $D = \max(N1,N2)$, then $D+3$ extended digits are used to evaluate the value $I_x(p,q)$. The three extra extended digits are used to guard against roundoff error. The range for which 70 significant decimal places can be achieved depends on the values p, q, and x .

5.8 Testing of the IBETA Program

To verify that desired accuracy is achieved by the program IBETA, two recurrence relationships of the incomplete beta function were used. They are

$$I_x(p,q) = xI_x(p-1,q) + (1-x)\ I_x(p,q-1)$$

and

$$I_x(p,q) = \frac{1}{p+q}\ \{pI_x(p+1,q) + qI_x(p,q+1)\}\ .$$

The following tables show the number of extended digits carried in the computation of $I_x(p,q)$ verses the number of decimal digits achieved. The number of decimal digits achieved using 10 extended digits is determined using one of the above recurrence relations.

Table 5.1 Number of correct significant digits achieved in the computation of $I_x(p,q)$ using ( ) extended digits

			X					
P	Q		.0001	.1000	.2000	.3000	.4000	.5000
10	10	(10)	29	29	30	30	31	30
		(7)	22	18	19	24	24	24
		(5)	14	11	11	12	10	10
100	10	(10)	30	30	31	31	30	30
		(7)	21	19	19	20	18	19
		(5)	14	14	11	12	12	11
100	100	(10)	30	31	31	31	30	30
		(7)	21	18	18	19	18	23
		(5)	14	13	12	10	9	15
1,000	10	(10)	29	30	32	29	30	32
		(7)	21	21	21	22	20	19
		(5)	13	13	12	12	15	14
1,000	100	(10)	29	30	31	31	30	30
		(7)	21	21	22	20	19	20
		(5)	11	13	13	11	12	11
1,000	1,000	(10)	29	31	30	30	30	30
		(7)	19	20	19	18	18	21
		(5)	12	11	13	10	10	13
10,000	10	(10)	30	31	29	31	31	30
		(7)	20	19	19	20	20	20
		(5)	12	11	11	12	13	13
10,000	100	(10)	31	29	30	30	31	31
		(7)	20	20	19	22	20	21
		(5)	12	11	11	12	12	12
10,000	1,000	(10)	30	30	30	31	29	30
		(7)	20	20	19	19	19	20
		(5)	12	11	10	13	13	13

					X		
P	Q		.6000	.7000	.8000	.9000	.9999
10	10	(10)	31	30	30	29	29
		(7)	24	24	19	18	22
		(5)	10	12	11	11	14
100	10	(10)	30	30	31	31	31
		(7)	23	23	23	23	20
		(5)	15	13	15	14	14
100	100	(10)	30	31	31	31	30
		(7)	18	19	18	18	21
		(5)	9	10	12	13	14
1,000	10	(10)	32	32	32	32	32
		(7)	24	23	23	23	21
		(5)	16	15	15	15	16
1,000	100	(10)	31	30	36	37	32
		(7)	24	22	23	21	21
		(5)	16	14	14	14	16
1,000	1,000	(10)	30	30	30	31	29
		(7)	19	20	22	21	21
		(5)	10	10	13	11	12
10,000	10	(10)	28	32	34	34	33
		(7)	20	20	20	19	19
		(5)	13	13	13	12	12
10,000	100	(10)	30	29	31	31	31
		(7)	20	20	19	18	20
		(5)	12	12	11	11	11
10,000	1,000	(10)	29	32	32	31	30
		(7)	21	20	19	21	20
		(5)	11	13	12	13	12

Table 5.1 (continued)

P	Q		X .0001	.1000	.2000	.3000	.4000	.5000
10,000	10,000	(10)	30	29	28	30	33	30
		(7)	19	19	20	22	20	20
		(5)	12	11	12	11	11	12
100,000	10	(10)	31	31	30	30	31	31
		(7)	18	19	19	20	20	19
		(5)	10	11	10	11	12	11
100,000	100	(10)	30	30	30	32	33	30
		(7)	19	19	19	19	20	18
		(5)	11	9	11	9	12	11
100,000	1,000	(10)	29	29	30	31	31	31
		(7)	18	19	18	18	18	19
		(5)	11	10	11	10	10	11
100,000	10,000	(10)	30	30	31	30	31	31
		(7)	18	19	19	19	20	20
		(5)	11	12	10	11	12	11
100,000	100,000	(10)	29	29	32	32	31	30
		(7)	19	19	19	20	19	20
		(5)	11	11	11	10	11	11
1,000,000	10	(10)	29	30	32	30	29	31
		(7)	18	19	18	18	19	19
		(5)	10	10	10	10	11	11
1,000,000	100	(10)	30	29	29	31	32	30
		(7)	18	18	18	17	20	20
		(5)	11	10	10	9	13	12
1,000,000	1,000	(10)	29	29	31	30	30	32
		(7)	18	18	18	18	18	19
		(5)	10	8	10	10	12	11
1,000,000	10,000	(10)	29	30	30	31	31	31
		(7)	17	18	19	18	19	20
		(5)	10	10	9	10	11	12
1,000,000	100,000	(10)	29	31	30	30	30	30
		(7)	18	17	18	18	20	20
		(5)	10	10	10	10	11	11

P	Q		X				
			.6000	.7000	.8000	.9000	.9999
10,000	10,000	(10)	33	30	28	29	30
		(7)	20	22	20	19	19
		(5)	11	11	12	11	12
100,000	10	(10)	31	32	31	30	31
		(7)	19	20	19	19	18
		(5)	11	12	11	11	11
100,000	100	(10)	32	32	31	33	31
		(7)	19	20	19	20	19
		(5)	11	11	11	13	11
100,000	1,000	(10)	32	32	32	31	33
		(7)	20	20	18	20	20
		(5)	12	12	12	11	12
100,000	10,000	(10)	30	32	31	33	30
		(7)	18	20	19	20	20
		(5)	11	11	11	12	11
100,000	100,000	(10)	31	32	32	29	29
		(7)	19	20	19	19	19
		(5)	11	10	11	11	11
1,000,000	10	(10)	32	31	34	32	27
		(7)	19	19	20	19	17
		(5)	12	11	12	11	12
1,000,000	100	(10)	32	32	32	32	33
		(7)	19	20	20	20	19
		(5)	11	12	10	12	12
1,000,000	1,000	(10)	30	32	32	32	32
		(7)	19	21	19	21	20
		(5)	12	12	12	11	12
1,000,000	10,000	(10)	30	31	33	31	30
		(7)	19	19	20	19	20
		(5)	12	10	12	11	10
1,000,000	100,000	(10)	30	31	29	33	31
		(7)	19	19	18	18	18
		(5)	11	10	11	11	11

Table 5.1 (continued)

P	Q		X .0001	.1000	.2000	.3000	.4000	.5000
1,000,000	1,000,000	(10)	30	30	30	33	32	30
		(7)	18	18	18	19	20	20
		(5)	11	12	9	10	11	12
10,000,000	10	(10)	30	31	27	28	31	28
		(7)	16	17	17	17	17	16
		(5)	8	9	8	8	9	8
10,000,000	100	(10)	29	29	29	29	28	28
		(7)	16	16	16	16	15	16
		(5)	8	8	8	8	6	8
10,000,000	1,000	(10)	30	30	31	30	31	31
		(7)	17	16	16	17	15	16
		(5)	8	8	8	8	8	7
10,000,000	10,000	(10)	28	29	29	29	29	28
		(7)	16	16	16	16	16	16
		(5)	8	8	8	7	8	8
10,000,000	100,000	(10)	28	28	29	28	29	28
		(7)	16	16	16	16	16	16
		(5)	8	7	9	8	8	8
10,000,000	1,000,000	(10)	28	29	29	28	29	29
		(7)	15	16	16	14	16	15
		(5)	8	9	9	8	10	10
10,000,000	10,000,000	(10)	28	29	28	29	28	28
		(7)	14	16	15	16	14	16
		(5)	8	8	7	8	7	8

			X				
P	Q		.6000	.7000	.8000	.9000	.9999
1,000,000	1,000,000	(10)	32	33	30	30	30
		(7)	20	19	18	18	18
		(5)	11	10	9	12	11
10,000,000	10	(10)	27	28	32	28	29
		(7)	14	16	16	15	16
		(5)	9	8	9	8	8
10,000,000	100	(10)	29	28	28	28	31
		(7)	16	16	16	16	16
		(5)	9	8	8	9	8
10,000,000	1,000	(10)	28	28	29	28	27
		(7)	16	16	16	15	16
		(5)	8	7	8	6	8
10,000,000	10,000	(10)	28	27	28	28	28
		(7)	16	15	16	15	16
		(5)	9	8	8	8	8
10,000,000	100,000	(10)	28	29	28	28	28
		(7)	15	16	14	16	15
		(5)	8	9	8	9	8
10,000,000	1,000,000	(10)	29	28	28	29	28
		(7)	16	15	15	16	15
		(5)	9	8	7	8	9
10,000,000	10,000,000	(10)	32	33	30	30	30
		(7)	20	19	18	18	18
		(5)	11	10	9	12	11

Table 5.2 Number of correct significant digits achieved in the computation of $I_x(p,q)$ using ( ) extended digits for small values of p, q, and x

P=1000	(10)	33	P=1000	(10)	33
Q=10	(7)	18	Q=10	(7)	18
X=.00001	(5)	12	X=.0000000000001	(5)	12
P=10	(10)	36	P=10	(10)	36
Q=10	(7)	23	Q=10	(7)	23
X=.00001	(5)	15	X=.0000000000001	(5)	15
P=1000	(10)	33	P=1000	(10)	31
Q=1000	(7)	20	Q=1000	(7)	20
X=.00001	(5)	13	X=.0000000000001	(5)	12

P=10	(10)	30	P=10	(10)	34	P=10	(10)	34
Q=.0001	(7)	23	Q=.00001	(7)	23	Q=.00001	(7)	22
X=.1	(6)	16	X=.00001	(6)	19	X=.0000000000001	(6)	20
	(5)	0		(5)	0		(5)	0
P=10	(18)	62	P=10	(18)	62	P=10	(18)	63
Q=.000000001	(13)	48	Q=.00000001	(13)	48	Q=.00000001	(13)	48
X=.1	(10)	37	X=.00001	(10)	37	X=.0000000000001	(10)	35
	(7)	0		(7)	0		(7)	0
	(5)	0		(5)	0		(5)	0

P=.000000001	(10)	36	P=.01	(10)	36
Q=10	(7)	23	Q=.001	(7)	23
X=.0000000000001	(5)	15	X=.00001	(5)	14

		X					
		.0001	.1	.2	.3	.4	.5
P=.1	(10)	33	30	30	31	31	31
Q=.1	(7)	24	22	21	21	21	21
	(5)	16	14	13	14	13	14

Table 5.3 The exponent of $I_x(p,q)$ if $x < \frac{p}{p+q}$, otherwise the exponent of $I_{1-x}(q,p)$

X=.0001

	Q				
P	10	100	1000	10000	100000
10	-35	-26	-16	-5	-1
100	-387	-341	-255	-158	-62
1000	-3978	-3856	-3600	-2547	-1568
10000	-39979	-39759	-38478	-33982	-25453
100000	-399960	-399660	-397567	-385448	-339801

X=.5

	Q				
P	10	100	1000	10000	100000
10	0	-20	-282	-2982	-30066
100	-20	0	-157	-2799	-29793
1000	-282	-157	0	-1858	-27968
10000	-2982	-2799	-1858	0	-18563
100000	-30066	-29793	-27968	-18563	0

6. INVERSE INCOMPLETE BETA FUNCTION

6.1 Existing Methods of Solution

Existing algorithms for finding percentage points of the beta distribution are based on use of the classical numerical root finding methods. Majumder and Bhattacharjee (1973b) and Roe (1969), use Newton's method. Given a starting value x_0 to the r percentage point x_r , Newton's method consists of using the iterative equation.

$$x_{i+1} = x_i - \frac{F(x_i)}{f(x_i)}$$

where

$$F(x_i) = r - I_{x_i}(p,q)$$

$$f(x_i) = F'(x_i) = - \frac{1}{B(p,q)} x^{p-1}(1-x)^{q-1} .$$

It should be noted that if x_0 is not sufficiently close to x_r , Newton's method will diverge.

Two other methods of computing the inverse incomplete beta function make use of the Illinois method and bisection. Boardman and Kopitzke use the Illinois method, which is a modification of the secant method. IMSL uses bisection to compute percentage points of the beta distribution for a certain parameter range. Otherwise, Newton's method is used. It should be remembered that all of these methods used to compute the inverse incomplete beta function need $I_x(p,q)$ evaluated accurately.

6.2 A New Method for Computation of the Inverse Incomplete Beta Function

This section will develop a new algorithm for the computation of percentage points of the beta distribution. The algorithm is based on results derived by Hill and Davis (1968). Suppose that $\{F_n(y)\}$ is a sequence of cdf's converging to a limiting distribution $F(y)$. If x and u are corresponding percentage points of F_n and F , respectively, then

$$F_n(x) = F(u)$$

Hill and Davis define u in terms of x and F .

Let

$$Z_n(x) = F_n(x) - F(x) .$$

Hence $$Z_n(x) = \int_x^u f(t)dt$$

where $f(t)$ is the pdf associated with F . Since,

$$\xi = \int_x^u f(t)dt$$

defines a relationship $u(\xi)$ with $u(0) = x$, $u(\xi)$ can be expanded in a Taylor's series around $\xi = 0$. Assuming that $f(u) \neq 0$.

$$\frac{du}{d\xi} = \frac{1}{f(u)}$$

and $$\frac{d^2u}{d\xi^2} = \frac{-f'(u)}{f^3(u)} = \frac{\psi(u)}{f^2(u)}$$

with $$\psi(u) = \frac{-f'(u)}{f(u)} \quad .$$

In general $$\frac{d^ku}{d\xi^k} = \frac{c_k(u)}{f^k(u)} ,$$

where $c_1(u) = 1$ and

$$c_{k+1}(u) = kc_k(u)\psi(u) + \frac{dc_k(u)}{du}$$

for $k = 1, 2, \ldots$.

Since $u(0) = x$, the Taylor's series is

$$u = x + \sum_{k=1}^{\infty} \frac{c_k(x)[Z_n(x_i)/f(x_i)]^k}{k!} \quad .$$

To derive the new method for computing the inverse incomplete beta function, consider the Taylor series derived by Hill and Davis. If the series is truncated at a finite number, an iterative method for finding approximations to u can be defined.

Let

$$x_{i+1} = x_i + \sum_{k=1}^{m} \frac{c_k(x_i)[Z_n(x_i)/f(x_i)]^k}{k!} \quad .$$

If m is chosen to be one, the method reduces to Newton's method.

Because the computation of the first three $c_k(x)$ is simple, the series was truncated with m equal to three.

$$c_1(x) = 1$$

$$c_2(x) = \frac{-f'(x)}{f(x)}$$

$$c_3(x) = 2\left(\frac{f'(x)}{f(x)}\right)^2 + \frac{d}{dx}\ \frac{-f'(x)}{f(x)}$$

For the beta distribution

$$f(x) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)}\ x^{p-1}(1-x)^{q-1} \quad .$$

Hence,

$$c_1(x) = 1$$

$$c_2(x) = -\ \left(\frac{p-1}{x} - \frac{q-1}{1-x}\right)$$

$$c_3(x) = 2\ \left(\frac{p-1}{x} - \frac{q-1}{1-x}\right)^2 + \frac{(p-1)}{x^2} + \frac{(q-1)}{(1-x)^2}$$

The iteration equation becomes

$$x_{i+1} = x_i + \frac{Z_n(x_i)}{f(x_i)} + \frac{c_2(x_i)}{2}\ \left(\frac{Z_n(x_i)}{f(x_i)}\right)^2 + \frac{c_3(x_i)}{6}\ \left(\frac{Z_n(x_i)}{f(x_i)}\right)^3 \quad .$$

An alternate way to derive the new method for computing the inverse incomplete beta function is based on a one point iteration formula given by Ralston (1965), in Section 8.4. The problem to be solved is to find a function F(x) such that

$$x_{i+1} = F(x_i)$$

and

$$\alpha = F(\alpha)$$

if the method converges. The function F(x) is used to find a root α of g(x) . The method is based on the Taylor series expansion of the inverse of g(x) in a neighborhood of α .

The series expansion has the form

$$\alpha = x_i - u_i \sum_{j=0}^{\infty} u_i^j Y_j \tag{6.1}$$

where $u_i = \frac{g(x_i)}{g'(x_i)}$

$$Y_0 = 1$$

and $Y_j = \frac{1}{j+1} \left(j \frac{g''(x_i)}{g'(x_i)} Y_{j-1} - Y'_{j-1}\right) \quad j > 0$.

Hill and Davis use the series expansion

$$\alpha = x_i + \sum_{k=1}^{\infty} \frac{c_k(x_i) \left[\frac{Z_n(x_i)}{f(x_i)}\right]^k}{k!} \tag{6.2}$$

where $c_1(x_i) = 1$

$$c_{k+1}(x_i) = k\, c_k(x_i)\psi(x_i) + \frac{d}{dx_i} c_k(x_i) \qquad k > 0$$

$$\psi(x_i) = \frac{-f'(x_i)}{f(x_i)}$$

and $Z_n(x_i) = F_n(x_i) - F(x_i) = r - F(x_i) = F(\alpha) - F(x_i)$.

In the equation (6.1), let $g(x) = r - I_x(p,q)$. Equation (6.1) can be shown to be equivalent to equation (6.2) as follows. First write

$$\alpha = x_i - u_i \sum_{j=0}^{\infty} u_i^j Y_j$$

Letting $Z_n(x) = r - I_x(p,q)$ and $f(x)$ be the density function of $I_x(p,q)$ equation (6.1) becomes

$$\alpha = x_i - \frac{Z_n(x_i)}{-f(x_i)} \sum_{j=0}^{\infty} \left(\frac{Z_n(x_i)}{-f(x_i)}\right)^j Y_j$$

or

$$\alpha = x_i + \sum_{j=0}^{\infty} \left(\frac{Z_n(x_i)}{f(x_i)}\right)^{j+1} (-1)^j Y_j \qquad (6.3)$$

It will be shown by induction that

$$(-1)^j Y_j = \frac{c_{j+1}(x_i)}{(j+1)!}$$

and hence, equation (6.1) is equivalent to equation (6.2).

$$(-1)^0 Y_0 = 1 = \frac{c_1(x_i)}{1!}$$

$$(-1)^1 Y_1 = (\frac{1}{2}) \frac{-f'(x_i)}{f(x_i)} = \frac{\psi(x_i)}{2!} = \frac{c_2(x_i)}{2!}$$

Assume $(-1)^j Y_j = \frac{c_{j+1}(x_i)}{(j+1)!}$.

By definition,

$$Y_{j+1} = \frac{1}{j+2} ((j+1) \frac{f'(x_i)}{f(x_i)} Y_j - \frac{d}{dx_i} Y_j)$$

So, $Y_{j+1} = \frac{1}{j+2} ((j+1) \frac{f'(x_i)}{f(x_i)} (-1)^j \frac{c_{j+1}(x_i)}{(j+1)!} - \frac{d}{dx_i} (-1)^j \frac{c_{j+1}(x_i)}{(j+1)!})$

Therefore,

$$(-1)^{j+1} Y_{j+1} = \frac{1}{j+2}((j+1) \frac{f'(x_i)}{f(x_i)} (-1) \frac{c_{j+1}(x_i)}{(j+1)!} - \frac{d}{dx_i} (-1) \frac{c_{j+1}(x_i)}{(j+1)!})$$

$$(-1)^{j+1} Y_{j+1} = \frac{1}{(j+2)!} ((j+1)\psi(x_i) c_{j+1}(x_i) + \frac{d}{dx_i} c_{j+1}(x_i))$$

$$(-1)^{j+1} Y_{j+1} = \frac{c_{j+2}(x_i)}{(j+2)!}$$

Hence, equation (6.1) is equivalent to equation (6.2).

Ralston has shown that if the iteration

$$x_{i+1} = F(x_i)$$

where

$$F(x_i) = x_i - u_i \sum_{j=0}^{m} u_i^j Y_j \qquad (6.4)$$

is used to solve for α, the method has order of convergence $m+2$. Using the fact that m is 2 in the new method, the rate of convergence of the new method is quartic. If m is equal to 0 then equation (6.4) is Newton's method. Table 6.1 shows the number of iterations needed to find the 95th percentile of the beta distribution using the new method and Newton's method. The initial starting value used was correct to two decimal places.

Table 6.1 Number of iterations needed to achieve a double precision result in finding the 95th percentile of beta (p,q) for the new method and Newton's method

	Q									
	10		50		100		150		200	
P	NEW	NEWTON	NEW	NEWTON	NEW	NEWTON	NEW	NEWTON	NEW	NEWTON
10	2	4	2	4	3	5	3	5	3	6
50	3	4	2	4	3	5	2	4	2	4
100	3	5	3	5	3	5	2	4	3	5
150	3	5	3	5	2	3	3	5	2	5
200	3	5	3	3	2	4	3	5	2	4

7. BIBLIOGRAPHY

Abramowitz, M., and I. Stegun, eds. 1964. Handbook of mathematical functions. National Bureau of Standards Applied Mathematics Series 55. U. S. Government Printing Office, Washington, D. C.

Aroian, L. A. 1941. Continued fractions for the incomplete beta functions. Ann. Math. Stat. 12:218-223.

Boardman, T. J. and R. W. Kopitzke. 1975. Probability and table values for statistical distributions. Proceedings of the Statistical Computing Section. ASA, Washington, D. C. 81-86.

Bouver, H., and R. E. Bargmann. 1975. Computational algorithms and modules for evaluation of statistical distribution functions. Themis Technical Report No. 36. University of Georgia, Athens, Georgia.

Brent, R. P. 1978. A Fortran multiple precision arithmetic package. ACM Transactions on Mathematical Software 4(1):57-70.

Cody, W. J., Jr. and W. Waite. 1980. Software manual for the elementary functions. Prentice-Hall, Englewood Cliffs, New Jersey. 269 pp.

Di Donato, A. R., and M. P. Jarnagin. 1967. The efficient calculation of the incomplete beta function ratio for half-integer values of parameters a,b. Math. Comp. 21:652-662.

Gautshi, W. 1964. Algorithm 222: Incomplete beta function ratios. Communications of the ACM 7(3):143-144.

Hill, G. W., and A. W. Davis. 1968. Generalized Asymptotic expansions of Cornish-Fisher type. Applied Math. Stat. 39(4):1264-1273.

Khovanskii, A. N. 1956. The applications of continued fractions and their generalizations to problems in approximation theory. Translated by Peter Wynn, 1963. P. Noordhoff, Groningen, Netherlands. 212 pp.

Knuth, D. E. 1981. The art of computer programming. Vol. 2. 2nd ed. Addison-Wesley Pub. Co., Reading, Mass. 680 pp.

Jones, W. B., and W. J. Thron. 1980. Continued fractions: Analytic theory and applications. Addison-Wesley Pub. Co., Reading, Mass. 428 pp.

Majumder, K. L., and G. P. Bhattacharjee. 1973a. Algorithm AS63: The incomplete beta integral. Applied Statistics 22(3):409-411.

Majumder, K. L., and G. P. Bhattacharjee, G. P. 1973b. Algorithm AS64: Inverse of the incomplete beta function. Appl. Stat. 22(3):411-414.

Miklosko, J. 1977. An algorithm for calculating continued fractions. Journal of Computational and Applied Mathematics 3(4):273-275.

Muir, T. 1876. New general formulae for the transformation of infinite series into continued fractions. Roy. Soc. Edinb. Trans. 27:467-469.

Muller, J. H. 1930. On the application of continued fractions to evaluation of certain integrals, with special reference to the incomplete beta function. Biometrika 22:284-297.

Osborn, D., and R. Madey. 1968. The incomplete beta function and its ratio to the complete beta function. Math. Comp. 22:159-162.

Peizer, D. B., and J. W. Pratt. 1968. A normal approximation for binomial, F, beta, and other common related tail probabilities. J. Am. Statist. Assoc. 63:1416-1456.

Ralston, A. 1965. A first course in numerical analysis. McGraw-Hill, New York. 578 pp.

Roe, G. M. 1969. Programs for the incomplete beta and gamma functions and their inverses. General Electric Research Report No. 69-C-045. General Electric, Louisville, KY.

Soper, H. E. 1921. The numerical evaluation of the incomplete B-function. Pages 1-49 in Tracts for Computers. Vol. 7. Cambridge Univ. Press, London, England.

Wall, H. S. 1948. Analytic theory of continued fractions. D. Van Nostrand, New York. 433 pp.

Wyatt, Jr., W. T. Lozier, and D. J. Orser. 1976. A portable extended precision arithmetic package and library with Fortran precompiler. ACM Transactions on Mathematical Software 2(3):209-231.

8. APPENDIX A

```
      SUBROUTINE IBETA1(P,Q,X,N,ANS)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C    THE SUBROUTINE IBETA1(P,Q,X,N,ANS) RETURNS THE VALUE I(P,Q;X)     C
C    IN THE DOUBLE PRECISION VARIABLE ANS, WITH N SIGNIFICANT          C
C    DECIMAL DIGITS. THE VARIABLE P, Q, AND X ARE DOUBLE PRECISION.    C
C    N IS AN INTEGER VARIABLE.                                         C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      REAL*8 P,Q,X,ANS,DIGIT
      DIMENSION S(3000)
C
C    DETERMINE THE NUMBER OF INTEGER VARIABLES NEEDED TO
C    CONTAIN A MP NUMBER WITH N DECIMAL DIGITS.
C
      ND=(N/4)+3
      PREC=ND
C
C    CONVERT P, Q, AND X TO MP NUMBERS.
C
      MPP=1
      MPQ=1+ND
      MPX=MPQ+ND
      MP1MX=MPX+ND
      CALL CONV(S(MPP),P,ND)
      CALL CONV(S(MPQ),Q,ND)
      CALL CONV(S(MPX),X,ND)
C
C    DETERMINE THE NUMBER OF EXTENDED DIGITS NEEDED TO
C    GUARANTY THAT THE RESULT WILL BE CORRECT TO N
C    SIGNIFICANT DECIMAL DIGITS.
C
      FQ=ND
      FP=ND
      FQX=ND
      FP1MX=ND
      IF (2*IABS(S(MPP+1)).GT.ND) FP=2*IABS(S(MPP+1))+3
      IF (2*IABS(S(MPQ+1)).GT.ND) FQ=2*IABS(S(MPQ+1))+3
      IF (IABS(S(MPQ+1)-S(MPX+1)).GT.ND) FQX=IABS(S(MPQ+1)-S(MPX+1))
      CALL ASSIGN(S(MP1MX),S(MPX),ND)
      S(MP1MX)=-1
      CALL ADD1(S(MP1MX),ND)
      FP1MX=IABS(S(MPP+1)-S(MP1MX))
      IF (FP1MX.LE.ND) FP1MX=ND
      MAXND=MAX0(FQ,FP,FQX,FP1MX)
      IF (MAXND.GT.ND) PREC=MIN0(MAXND,31)
C
C    FIND THE MP VALUES OF P, Q, X, 1-X, AND P+Q WITH
C    SUFFICIENT MP DIGITS SO THAT THE RESULT IS ACCURATE
C    TO N SIGNIFICANT DECIMAL DIGITS.
C
      PREC=PREC+3
      MPQ=1+PREC
      MPX=MPQ+PREC
      MP1MX=MPX+PREC
      MPANS=MP1MX+PREC
      MPPQ=MPANS+PREC
      MPMEAN=MPPQ+PREC
      MPX12=MPMEAN+PREC
      WORK=MPX12+PREC
      CALL CONV(S(MPP),P,PREC)
      CALL CONV(S(MPQ),Q,PREC)
      CALL CONV(S(MPX),X,PREC)
      CALL ADD(S(MPP),S(MPQ),S(MPPQ),PREC)
      CALL ASSIGN(S(MP1MX),S(MPX),PREC)
      S(MP1MX)=-1
      CALL ADD1(S(MP1MX),PREC)
C
C    CALL THE SUBROUTINE CASE TO SOLVE I(P,Q;X).
C
      CALL CASE(S(MPP),S(MPQ),S(MPX),S(MPPQ),S(MP1MX),S(MPANS),
     1S(WORK),PREC,3000-WORK,P,Q,P+Q,S(MPMEAN),S(MPX12))
C
C    CONVERT THE MP RESULT TO DOUBLE PRECISION.
C
      POWER=S(MPANS+1)
      ANS=0.D0
C
C    EXPONENT OVERFLOW OCCURRED. THE NUMBER IS TOO SMALL FOR
C    MP REPRESENTATION. RETURN THE VALUE AS ZERO.
C
      IF (POWER.GT.0) RETURN
C
C    THE MP RESULT IS TOO SMALL TO REPRESENT AS AN IBM
C    DOUBLE PRECISION NUMBER. RETURN THE VALUE AS ZERO.
C    ON NON-IBM MACHINES THIS LOWER BOUND MAY NEED TO
C    BE CHANGED.
C
      IF (POWER.LT.-18) RETURN
      MANT=ND-1
C
C     CONVERT THE MP MANTISSA TO DOUBLE PRECISION.
C
      DO 10 K=1,MANT
         DIGIT=-DFLOAT(MANT-K+1)
         ANS=ANS+S(MPANS+ND+1-K)*10000.D0**DIGIT
 10   CONTINUE
C
C     CONVERT THE MP EXPONENT
C
      ANS=ANS*10000.D0**DFLOAT(POWER)
      RETURN
      END
      SUBROUTINE IBETA2(P,Q,X,N,S,ANS)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C     THE SUBROUTINE IBETA2(P,Q,X,N,S,ANS) COMPUTES THE VALUE OF       C
C     I(P,Q;X) AND RETURNS IT AS A MP NUMBER IN THE INTEGER ARRAY      C
C     ANS(S). THE VARIABLES P, Q, AND X ARE DOUBLE PRECISION. N IS     C
C     AN INTEGER VARIABLE SPECIFING THE NUMBER OF SIGNIFICANT DECIMAL  C
C     DIGITS TO BE CONTAINED IN THE RESULT. S IS AN INTEGER            C
C     VARIABLE AND MUST HAVE THE VALUE INT(N/4)+3.                     C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      REAL*8 P,Q,X,ANS
      DIMENSION W(3000),ANS(S)
C
C     DETERMINE THE NUMBER OF INTEGER VARIABLES NEEDED TO
C     CONTAIN A MP NUMBER WITH N DECIMAL DIGITS.
C
      ND=(N/4)+3
      PREC=ND
C
C     CONVERT P, Q, AND X TO MP NUMBERS.
C
      MPP=1
      MPQ=1+ND
      MPX=MPQ+ND
      MP1MX=MPX+ND
      CALL CONV(W(MPP),P,ND)
      CALL CONV(W(MPQ),Q,ND)
      CALL CONV(W(MPX),X,ND)
C
C     DETERMINE THE NUMBER OF EXTENDED DIGITS NEEDED TO
C     GUARANTY THAT THE RESULT WILL BE CORRECT TO N
C     SIGNIFICANT DECIMAL DIGITS.
C
      FQ=ND
      FP=ND
      FQX=ND
      FP1MX=ND
      IF (2*IABS(W(MPP+1)).GT.ND) FP=2*IABS(W(MPP+1))+3
      IF (2*IABS(W(MPQ+1)).GT.ND) FQ=2*IABS(W(MPQ+1))+3
      IF (IABS(W(MPQ+1)-W(MPX+1)).GT.ND) FQX=IABS(W(MPQ+1)-W(MPX+1))
      CALL ASSIGN(W(MP1MX),W(MPX),ND)
      W(MP1MX)=-1
      CALL ADD1(W(MP1MX),ND)
      FP1MX=IABS(W(MPP+1)-W(MP1MX))
      IF (FP1MX.LE.ND) FP1MX=ND
      MAXND=MAX0(FQ,FP,FQX,FP1MX)
      IF (MAXND.GT.ND) PREC=MIN0(MAXND,31)
C
C     FIND THE MP VALUES OF P, Q, X, 1-X, AND P+Q WITH
C     SUFFICIENT MP DIGITS SO THAT THE RESULT IS ACCURATE
C     TO N SIGNIFICANT DECIMAL DIGITS.
C
      PREC=PREC+3
      MPQ=1+PREC
      MPX=MPQ+PREC
      MP1MX=MPX+PREC
      MPANS=MP1MX+PREC
      MPPQ=MPANS+PREC
      MPMEAN=MPPQ+PREC
      MPX12=MPMEAN+PREC
      WORK=MPX12+PREC
      CALL CONV(W(MPP),P,PREC)
      CALL CONV(W(MPQ),Q,PREC)
      CALL CONV(W(MPX),X,PREC)
      CALL ADD(W(MPP),W(MPQ),W(MPPQ),PREC)
      CALL ASSIGN(W(MP1MX),W(MPX),PREC)
      W(MP1MX)=-1
      CALL ADD1(W(MP1MX),PREC)
C
C     CALL THE SUBROUTINE CASE TO SOLVE I(P,Q;X).
C
      CALL CASE(W(MPP),W(MPQ),W(MPX),W(MPPQ),W(MP1MX),W(MPANS),
     1W(WORK),PREC,3000-PREC,P,Q,P+Q,W(MPMEAN),W(MPX12))
      CALL ASSIGN(ANS,W(MPANS),S)
      IF (ANS(2).GE.0) ANS(1)=0
      RETURN
      END
```

```
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C                                                                     C
C    THE SUBROUTINE IBETA3(P,Q,X,N,ANS) FINDS THE VALUE OF I(P,Q;X)    C
C    AND RETURNS THE VALUE IN THE ARRAY ANS(N). P, Q, X, AND ANS      C
C    ARE INTEGER ARRAYS CONTAINING MP NUMBERS WITH N-2 EXTENDED       C
C    DIGITS. THE VALUE I(P,Q;X) IS COMPUTED TO N-2 SIGNIFICANT        C
C    EXTENDED DIGITS.                                                 C
C                                                                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE IBETA3(P,Q,X,N,ANS)
      IMPLICIT INTEGER (A-Z)
      REAL*8 PR,QR,DIGIT
      DIMENSION S(3000),P(N),Q(N),X(N),ANS(N)
      PREC=N
C
C    DETERMINE THE NUMBER OF EXTENDED DIGITS NEEDED TO
C    GUARANTY THAT THE RESULT WILL BE CORRECT TO N-2
C    SIGNIFICANT EXTENDED DIGITS.
C
      MP1MX=1
      FQ=N
      FP=N
      FQX=N
      FP1MX=N
      IF (2*IABS(P(2)).GT.N) FP=2*IABS(P(2))+3
      IF (2*IABS(Q(2)).GT.N) FQ=2*IABS(Q(2))+3
      IF (IABS(Q(2)-X(2)).GT.N) FQX=IABS(Q(2)-X(2))
      CALL ASSIGN(S(MP1MX),X,N)
      S(MP1MX)=-1
      CALL ADD1(S(MP1MX),N)
      FP1MX=IABS(P(2)-S(MP1MX))
      IF (FP1MX.LE.N) FP1MX=N
      MAXN=MAX0(FQ,FP,FQX,FP1MX)
      IF (MAXN.GT.N) PREC=MIN0(MAXN,31)
C
C    FIND THE MP VALUES OF P, Q, X, 1-X, AND P+Q WITH
C    SUFFICIENT MP DIGITS SO THAT THE RESULT IS ACCURATE
C    TO N-2 SIGNIFICANT EXTENDED DIGITS.
C
      MPP=1
      PREC=PREC+3
      MPQ=1+PREC
      MPX=MPQ+PREC
      MP1MX=MPX+PREC
      MPANS=MP1MX+PREC
      MPPQ=MPANS+PREC
      MPMEAN=MPPQ+PREC
      MPX12=MPMEAN+PREC
      WORK=MPX12+PREC
      CALL ASSIGN(S(MPP),P,N)
      CALL ASSIGN(S(MPQ),Q,N)
      CALL ASSIGN(S(MPX),X,N)
      N1=N+1
      PREC1=PREC-1
      DO 10 K=N1,PREC1
         S(MPP+K)=0
         S(MPQ+K)=0
         S(MPX+K)=0
10    CONTINUE
       POWER=P(2)
       PR=0.D0
C
C    FIND THE DOUBLE PRECISION VALUE OF THE MP NUMBER P
C    TO BE USED BY THE LGAM SUBROUTINE.
C
       IF (POWER.GE.18) PR=.1D72
       IF (POWER.LE.-18) PR=.1D-72
       IF (PR.NE.0.D0) GOTO 30
       ND=MIN0(N,5)
       MANT=ND-1
       DO 20 K=1,MANT
       DIGIT=-DFLOAT(MANT-K+1)
       PR=PR+DFLOAT(S(MPP+ND+1-K))*10000.D0**DIGIT
20     CONTINUE
       PR=PR*10000.D0**DFLOAT(POWER)
C
C    FIND THE DOUBLE PRECISION VALUE OF THE MP NUMBER Q
C    TO BE USED BY THE LGAM SUBROUTINE.
C
30     POWER=Q(2)
       QR=0.D0
       IF (POWER.GE.18) QR=.1D72
       IF (POWER.LE.-18) QR=.1D-72
       IF (QR.NE.0.D0) GOTO 50
       ND=MIN0(N,5)
       MANT=ND-1
       DO 40 K=1,MANT
       DIGIT=-DFLOAT(MANT-K+1)
       QR=QR+DFLOAT(S(MPQ+ND+1-K))*10000.D0**DIGIT
40     CONTINUE
       QR=QR*10000.D0**DFLOAT(POWER)
50    CALL ADD(S(MPP),S(MPQ),S(MPPQ),PREC)
      CALL ASSIGN(S(MP1MX),S(MPX),PREC)
      S(MP1MX)=-1
      CALL ADD1(S(MP1MX),PREC)
```

```
C
C     CALL CASE TO SOLVE I(P,Q;X)
C
      CALL CASE(S(MPP),S(MPQ),S(MPX),S(MPPQ),S(MP1MX),S(MPANS),
     1S(WORK),PREC,3000-WORK,PR,QR,PR+QR,S(MPMEAN),S(MPX12))
      CALL ASSIGN(ANS,S(MPANS),N)
      RETURN
      END
      SUBROUTINE ADD(X,Y,Z,P)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C    THE SUBROUTINE ADD(X,Y,Z,P) ADDS THE MP NUMBERS X AND Y. THE      C
C    SUM IS RETURNED IN THE MP NUMBER Z. THREE OTHER SUBROUTINES       C
C    ARE USED TO PERFORM CERTAIN TYPES OF ADDITION. THE SUBROUTINE     C
C    UNADD(X,Y,Z,P,NORM) PERFORMS ADDITION WHEN X AND Y HAVE           C
C    DIFFERENT SIGNS AND UNEQUAL EXPONENTS. IF X AND  Y HAVE THE SAME  C
C    SIGN BUT DIFFERENT EXPONENTS THE SUBROUTINE EADD(X,Y,Z,P,NORM)    C
C    IS USED. TO ADD NUMBERS WITH THE SAME EXPONENT BUT DIFFERENT      C
C    SIGNS THE SUBROUTINE UADD(X,Y,Z,P) IS CALLED. THE SUBROUTINE      C
C    ADD PERFORMS ADDITION ON MP NUMBERS WITH THE SAME SIGN AND        C
C    EXPONENT. OTHERWISE ADD CALLS ONE OF THE ABOVE SUBROUTINES        C
C    TO PERFORM ADDITION.                                              C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IMPLICIT INTEGER (A-Z)
      DIMENSION X(P),Y(P),Z(P)
C
C    IF EITHER X OR Y IS ZERO, SET Z TO THE APPROPRIATE VALUE AND
C    RETURN.
C
      IF (X(1).NE.0) GOTO 10
        CALL ASSIGN(Z,Y,P)
        RETURN
10    IF (Y(1).NE.0) GOTO 20
        CALL ASSIGN(Z,X,P)
        RETURN
C
C   FIND THE DIFFERENCE OF THE EXPONENTS.
C
20    NORM=X(2)-Y(2)
      P3=P-2
      IF (NORM) 200,30,300
C
C    THE NUMBERS X AND Y HAVE THE SAME EXPONENT. DETERMINE
C    IF THEY HAVE THE SAME SIGN.
C
30    IF (X(1).NE.Y(1)) GOTO 60
        POS=P+1
        ICARRY=0
```

```
C
C     X AND Y HAVE THE SAME SIGN. ADDITION IS PERFORMED
C     BY LOOPING UNTIL ALL DIGITS AND CARRIES HAVE BEEN
C     COMPUTED. IF THE LAST ADDITION GENERATES A CARRY
C     THE EXPONENT OF Z IS ADJUSTED. ALSO THE DIGITS ARE
C     SHIFTED AND THE LEADING DIGIT IS ASSIGNED THE VALUE
C     1.
C
         DO 40 J=1,P3
           POS1=POS-J
           ITOT=X(POS1)+Y(POS1)+ICARRY
           IF (ITOT.LT.10000) GOTO 35
             ICARRY=1
             Z(POS1)=ITOT-10000
             GOTO 40
35         Z(POS1)=ITOT
           ICARRY=0
40       CONTINUE
         IF (ICARRY.NE.0) GOTO 45
           Z(1)=X(1)
           Z(2)=X(2)
           RETURN
45       P31=P3-1
         DO 50 J=1,P31
           POS1=POS-J
           Z(POS1)=Z(POS1-1)
50       CONTINUE
         Z(3)=1
         Z(2)=X(2)+1
         Z(1)=X(1)
         RETURN
C
C     X AND Y HAVE THE SAME EXPONENT BUT DIFFERENT SIGNS. DETERMINE
C     WHICH OF X OR Y HAS THE LARGER ABSOLUTE VALUE. THEN CALL
C     EADD WITH THE PARAMETERS X AND Y ORDERED SO THAT THE NUMBER
C     WITH THE SMALLER ABSOLUTE VALUE IS SUBTRACTED FROM THE
C     NUMBER WITH THE LARGER ABSOLUTE VALUE.
C
60     DO 70 J=3,P
         IF (X(J)-Y(J)) 90,70,80
70     CONTINUE
       Z(1)=0
       RETURN
80     CALL EADD(X,Y,Z,P)
       RETURN
90     CALL EADD(Y,X,Z,P)
       RETURN
```

```
C
C     Y IS LARGER THAN X IN ABSOLUTE VALUE. CALL EITHER
C     UADD OR UNADD AS THE SIGNS OF X AND Y DICTATE. THE ORDER
C     OF X AND Y IN THE CALL STATEMENTS IS SUCH THAT THE NUMBER
C     WITH THE LARGER ABSOLUTE VALUE IS FIRST.
C
200    IF (X(1).EQ.Y(1)) GOTO 210
         CALL UNADD(Y,X,Z,P,NORM)
         RETURN
210    CALL UADD(Y,X,Z,P,NORM)
       RETURN
C
C     X IS LARGER THAN Y IN ABSOLUTE VALUE. CALL UADD OR UNADD
C     AS THE SIGNS OF X AND Y INDICATE.
C
300    IF (X(1).EQ.Y(1)) GOTO 310
         CALL UNADD(X,Y,Z,P,-NORM)
         RETURN
310    CALL UADD(X,Y,Z,P,-NORM)
       RETURN
       END
```

```
      SUBROUTINE ADD1(X,P)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C                                                                     C
C    THE SUBROUTINE ADD1(X,P) ADDS ONE TO THE VALUE OF THE MP         C
C    NUMBER X AND RETURNS THE SUM IN X.                               C
C                                                                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION X(P)
      M=X(2)
C
C    DETERMINE IF ADDITION OR SUBTRACTION IS TO BE PERFORMED.
C
      IF(X(1))30,10,230
C
C    THE NUMBER X HAS THE VALUE ZERO. SET X TO ONE AND RETURN.
C
10    X(3)=1
      X(2)=1
      X(1)=1
      DO 20 J=4,P
        X(J)=0
20    CONTINUE
      RETURN
C
C    SUBTRACTION IS TO BE PERFORMED. DETERMINE THE RELATIVE
C    MAGNITUDE OF THE MP NUMBER X TO 1.
C
30    IF (M.GT.0) GOTO 190
C
C    THE ABSOLUTE VALUE OF X IS LESS THAN 1.
C
        IF (P-3+M) 40,60,110
C
C    THE VALUE OF X IS SO SMALL THAT THE VALUE OF X=.999999......
C
40          X(1)=1
            X(2)=0
            DO 50 J=3,P
              X(J)=9999
50          CONTINUE
```

```
          RETURN
C
C    ONLY THE FIRST DIGIT OF X IS NEEDED TO PERFORM THE
C    SUBTRACTION PLUS A GAURD VALUE.
C
60        G=X(3)
          DO 70 J=4,P
            IF (X(J).NE.0) GOTO 80
70        CONTINUE
          GOTO 90
80        G=G+1
90        X(1)=1
          X(2)=0
          P1=P-1
          DO 100 J=3,P1
            X(J)=9999
100       CONTINUE
          X(P)=10000-G
          RETURN
C
C    SEVERAL DIGITS OF X ARE NEEDED TO PERFORM THE SUBTRACTION.
C    NOTE THAT A BORROW IS GENERATED IN EVERY SUBTRACTION. HENCE
C    THE FILLING DIGITS OF THE SUM ARE 9999, IF A DIGIT OF X IS
C    NOT USED TO DETERMINE THE VALUE OF THE DIGIT.
C
110     XP=P+M
        G=X(XP)
        IF (M.EQ.0) GOTO 140
          XP1=XP+1
          DO 120 J=XP1,P
            IF (X(J).NE.0) GOTO 130
120       CONTINUE
          GOTO 140
130       G=G+1
140       IBRW=0
          IF (G.EQ.0) GOTO 145
            IBRW=1
            G=10000-G
145       XP2=XP+2
          P2=P+2
          XP1=XP-1
          DO 160 J=3,XP1
            POS=XP2-J
            POS2=P2-J
            ITOT=10000-X(POS)-IBRW
            IF (ITOT.EQ.10000) GOTO 150
              IBRW=1
              X(POS2)=ITOT
              GOTO 160
150         IBRW=0
```

```
          X(POS2)=0
160     CONTINUE
        X(P)=G
        IF (POS2.EQ.3) GOTO 180
          POS21=POS2-1
          DO 170 J=3,POS21
            X(J)=9999
170       CONTINUE
180     X(1)=1
        X(2)=0
        CALL VNORM(X,P)
        RETURN
C
C    THE ABSOLUTE VALUE OF X IS GREATER THAN 1. FIND THE DIGIT
C    OF X IN WHICH SUBTRACTION BEGINS. CONTINUE SUBTRACTING UNTIL
C    A BORROW IS NOT GENERATED. NORMALIZE THE RESULT AND RETURN.
C
190   IF (P-2.LT.M) M=P-2
      M3=3+M
      DO 200 J=1,M
        POS=M3-J
        ITOT=X(POS)-1
        X(POS)=ITOT
        IF (ITOT.GE.0) GOTO 210
          X(POS)=9999
200   CONTINUE
210   IF (POS.GT.3) RETURN
      IF (X(3).NE. 0) RETURN
        P1=P-1
        DO 220 J=3,P1
           X(J)=X(J+1)
220     CONTINUE
        X(2)=X(2)-1
        X(P)=0
        CALL VNORM(X,P)
        RETURN
C
C    ADDITION IS THE OPERATION TO BE PERFORMED.
C
230   IF (M.LT.1) GOTO 260
C
C    THE NUMBER X IS SO LARGE THE ADDITION OF 1 WILL NOT
C    CHANGE THE VALUE OF X. HENCE RETURN.
C
      IF (M.GT.P-2) RETURN
```

```
C     THE VALUE OF X IS LARGER THAN ONE. FIND THE DIGIT OF X
C     WHERE THE ADDITION IS TO BEGIN. CONTINUE ADDITION UNTIL
C     A ZERO CARRY IS GENERATED. ADJUST THE REPRESENTATION OF
C     X TO THE CORRECT FORM AND RETURN.
C
        M3=3+M
        DO 240 J=1,M
          POS=M3-J
          ITOT=X(POS)+1
          X(POS)=ITOT
          IF (ITOT.LT.10000) RETURN
            X(POS)=0
240     CONTINUE
        P3=P+3
        P1=P-1
        DO 250 J=3,P1
           POS=P3-J
           X(POS)=X(POS-1)
250     CONTINUE
        X(3)=1
        X(2)=X(2)+1
        RETURN
C
C    THE VALUE OF X IS LESS THAN 1. IF X IS SUFFICIENTLY SMALL
C    RETURN THE VALUE OF X AS 1.
C
260   IF (-M.GE.P-3) GOTO 10
C
C    SHIFT THE DIGITS OF X SO THAT WHEN THE EXPONENT IS SET TO
C    1 AND THE LEADING DIGIT SET TO 1, THE VALUE OF X HAS BECOME
C    1+X.
C
        NM= P-3+M
        P2=P+M
        P1=P+1
        DO 270 J=1,NM
          POS=P1-J
          POS2=P2-J
          X(POS)=X(POS2)
270     CONTINUE
        X(1)=1
        X(2)=1
        X(3)=1
        IF (POS.EQ.4) RETURN
          POS1=POS-1
          DO 280 J=4,POS1
            X(J)=0
280       CONTINUE
          RETURN
      END
      SUBROUTINE ASSIGN(X,Y,P)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C   THE SUBROUTINE ASSIGN(X,Y,P) ASSIGNS THE MP NUMBER X THE VALUE     C
C   OF THE MP NUMBER Y.                                                C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      INTEGER X,Y,P
      DIMENSION X(P),Y(P)
      DO 10  J=1,P
        X(J)=Y(J)
10    CONTINUE
      RETURN
      END
      SUBROUTINE CASE(P,Q,X,PQ,X1,Z,W,D,M,PX,QX,PQX,MEAN,X12)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C                                                                       C
C    THE SUBROUTINE CASE DETERMINES WHICH METHOD TO USE IN FINDING      C
C    THE VALUE OF I(P,Q;X). THE SUBROUTINE MULLER SOLVES FOR THE        C
C    VALUE I(P,Q;X) USING MULLER'S CONTINUED FRACTION. THE              C
C    SUBROUTINE RPLQ USES THE CONTINUED FRACTIONS FOR THE RECURRENCE    C
C    RELATIONS : I(P,Q;X)=B(P,Q)*(X**P)*((1-X)**(Q-1))*(1/P)            C
C                              + I(P+1,Q-1;X)                           C
C          AND                                                          C
C                I(P,Q;X)=B(P,Q)*(X**P)*((1-X)**Q))*(1/P)               C
C                              + I(P+1,Q;X)                             C
C          , WHERE B(P,Q) IS THE COMPLETE BETA FUNCTION.                C
C                                                                       C
C    WHEN X>P/(P+Q)=MEAN THE IDENTITY                                   C
C                I(P,Q;X)=1-I(Q,P;1-X)                                  C
C    , IS USED TO SOLVE FOR I(P,Q,X).                                   C
C    MULLER'S CONTINUED FRACTION IS EMPLOYED WHEN X=<1/2 AND            C
C    X=<P/(P+Q)=MEAN, OR X>1/2 AND X>MEAN. OTHERWISE, THE               C
C    SUBROUTINE RPLQ IS USED.                                           C
C                                                                       C
C    ARGUMENT     DESCRIPTION                                           C
C       P         PARAMETER P IN I(P,Q;X)                               C
C       Q         PARAMETER Q IN I(P,Q;X)                               C
C       X         ARGUEMENT X IN I(P,Q;X)                               C
C       PQ        P+Q                                                   C
C       X1        1-X                                                   C
C       Z         RETURNED VALUE OF I(P,Q;X)                            C
C       W         WORK SPACE                                            C
C       D         DIMENSION OF MP NUMBERS                               C
C       M         DIMENSION OF WORK SPACE                               C
C       PX        DOUBLE PRECISION VALUE OF P                           C
C       QX        DOUBLE PRECISION VALUE OF Q                           C
C       PQX       DOUBLE PRECISION VALUE OF PQ                          C
C       MEAN      SPACE FOR THE MP MEAN OF I(P,Q;X)                     C
C       X12       MP VALUE 1/2                                          C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      REAL*8 PX,QX,PQX
      DIMENSION P(D),Q(D),X(D),X1(D),PQ(D),Z(D),W(M),MEAN(D),X12(D)
C
C    SET UP THE MP NUMBER CONTAINING 1/2
C
      DO 10 I=4,D
        X12(I)=0
```

```
10     CONTINUE
       X12(1)=-1
       X12(2)=0
       X12(3)=5000
C
C     FIND IF X=<1/2
C
       CALL ADD(X12,X,MEAN,D)
       IF  (MEAN(1).EQ.1) GOTO 200
C
C     X=<1/2 FIND IF X=<P/(P+Q)=MEAN
C
         CALL DIV(P,PQ,MEAN,W,D,M)
         MEAN(1)=-1
         CALL ADD(X,MEAN,MEAN,D)
         IF (MEAN(1).EQ.1) GOTO 100
C
C     X=<1/2 AND X<MEAN USE SUBROUTINE MULLER TO FIND I(P,Q;X)
C
           CALL MULLER(P,Q,X,PQ,X1,Z,W,D,M,PX,QX,PQX)
           RETURN
C
C     X=<1/2 AND X>MEAN USE SUBROUTINE RPLQ TO FIND I(Q,P;1-X)
C     RETURN THE VALUE 1-I(Q,P;1-X)
C
100      CALL RPLQ(Q,P,X1,PQ,X,Z,W,D,M,QX,PX,PQX)
         Z(1)=-1
         CALL ADD1(Z,D)
         RETURN
C
C     X>1/2 FIND IF X=<MEAN
C
200     CALL DIV(P,PQ,MEAN,W,D,M)
        MEAN(1)=-1
        CALL ADD(X,MEAN,MEAN,D)
        IF (MEAN(1).EQ.1) GOTO 300
C
C     X>1/2 AND X=<MEAN USE SUBROUTINE RPLQ TO FIND I(P,Q;X)
C
          CALL RPLQ(P,Q,X,PQ,X1,Z,W,D,M,PX,QX,PQX)
          RETURN
C
C     X>1/2 AND X>MEAN USE SUBROUTINE MULLER TO FIND I(Q,P;1-X)
C     RETURN THE VALUE 1-I(Q,P;1-X)
C
300     CALL MULLER(Q,P,X1,PQ,X,Z,W,D,M,QX,PX,PQX)
        Z(1)=-1
        CALL ADD1(Z,D)
        RETURN
        END
```

```
      SUBROUTINE CONV(Y,X,P)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C    THE SUBROUTINE CONV(Y,X,P) CONVERTS THE IBM DOUBLE PRECISION      C
C    VARIABLE X TO THE MP NUMBER Y. THE MP NUMBER Y CONTAINS P-2       C
C    EXTENDED DIGITS.                                                  C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      INTEGER Y,P,Z,R,POW,SUM,MSUM,EXPON,SUM1,M1,M2
      REAL*8 X,X1
      DIMENSION Y(P),Z(15),R(15,17),POW(2),SUM1(17),M1(84),M2(84)
      EQUIVALENCE (X1,POW(1))
      Y(1)=1
      M1(1)=0
C
C    FIND THE SIGN.
C
      IF (X)20,10,30
10      Y(1)=0
      RETURN
20      X=-X
        Y(1)=-1
C
C   INITIALIZE THE VALUES OF 16**(-K) FOR K=1,15 IN MP NUMBERS
C   CONTAINING SUFFICIENT DIGITS TO REPRESENT 16**(-K)
C   EXACTLY. FOR MACHINES OTHER THAN THOSE THAT REPRESENT
C   DOUBLE PRECISION NUMBERS IN 64 EXCESS FORMWITH 15 HEX DIGITS,
C   ADDITIONAL VALUES OF 16**(-K) FOR K=16,.. MIGHT BE NEEDED. THE
C   VALUES OF 16**(-K) ARE USED TO CONVERT THE MANTISSA OF
C   THE DOUBLE PRECISION NUMBER X.
C
30    DO 50 I=1,15
        R(I,1)=1
        DO 40 J=3,17
          R(I,J)=0
40      CONTINUE
50    CONTINUE
      R(1,2)=0
      R(1,3)=625
      R(2,2)=0
      R(2,3)=39
      R(2,4)=625
      R(3,2)=0
      R(3,3)=2
      R(3,4)=4414
      R(3,5)=625
```

```
 R(4,2)=-1
 R(4,3)=1525
 R(4,4)=8789
 R(4,5)=625
 R(5,2)=-1
 R(5,3)=95
 R(5,4)=3674
 R(5,5)=3164
 R(5,6)=625
 R(6,2)=-1
 R(6,3)=5
 R(6,4)=9604
 R(6,5)=6447
 R(6,6)=7539
 R(6,7)=625
 R(7,2)=-2
 R(7,3)=3725
 R(7,4)=2902
 R(7,5)=9846
 R(7,6)=1914
 R(7,7)=625
 R(8,2)=-2
 R(8,3)=232
 R(8,4)=8306
 R(8,5)=4365
 R(8,6)=3869
 R(8,7)=6289
 R(8,8)=625
 R(9,2)=-2
 R(9,3)=14
 R(9,4)=5519
 R(9,5)=1522
 R(9,6)=8366
 R(9,7)=8518
 R(9,8)=664
 R(9,9)=625
 R(10,2)=-3
 R(10,3)=9094
 R(10,4)=9470
 R(10,5)=1772
 R(10,6)=9282
 R(10,7)=3791
R(10,8)=5039
R(10,9)=625
 R(11,2)=-3
 R(11,3)=568
 R(11,4)=4341
 R(11,5)=8860
 R(11,6)=8080
 R(11,7)=1486
```

```
R(11,8)=9689
R(11,9)=9414
R(11,10)=625
 R(12,2)=-3
 R(12,3)=035
 R(12,4)=5271
 R(12,5)=3678
 R(12,6)=8005
 R(12,7)=926
R(12,8)=9355
R(12,9)=6213
R(12,10)=3789
 R(12,11)=625
 R(13,2)=-3
 R(13,3)=002
 R(13,4)=2204
 R(13,5)=4604
 R(13,6)=9250
R(13,7)=3130
R(13,8)=8084
R(13,9)=7263
R(13,10)=3361
 R(13,11)=8164
R(13,12)=0625
 R(14,2)=-4
 R(14,3)=1387
 R(14,4)=7787
 R(14,5)=8078
 R(14,6)=1445
R(14,7)=6755
R(14,8)=2953
R(14,9)=9585
R(14,10)=1135
 R(14,11)=2539
R(14,12)=0625
 R(15,2)=-4
 R(15,3)=0086
 R(15,4)=7361
 R(15,5)=7379
 R(15,6)=8840
R(15,7)=3547
R(15,8)=2059
R(15,9)=6224
R(15,10)=0695
 R(15,11)=9533
R(15,12)=6914
R(15,13)=625
```

```
C
C   FIND THE EXPONENT IN 64 EXCESS NOTATION. FOR OTHER MACHINES
C   THE CONSTANT 16777216 SHOULD BE REPLACED WITH THE INTEGER
C   SUCH THAT INTEGER DIVISION BY THE NUMBER CAUSES A RIGHT
C   SHIFT LEAVING ONLY THE EXPONENT BITS IN EXPON.
C
      X1=X
      EXPON=POW(1)/16777216
      EXPON=EXPON-64
      IWIPE=-EXPON*16777216
C
C   FIND THE MANTISSA BITS.
C
      POW(1)=POW(1)+IWIPE
C
C    CONVERT THE MANTISSA TO A MP NUMBER.
C
      DO 60 I=1,15
        Z(I)=IFIX(SNGL(X1*16.D0))
        X1=X1-DFLOAT(Z(I))/16.D0
        X1=X1*16.D0
60    CONTINUE
      DO 80 I=1,15
        IF (Z(I).EQ.0) GOTO 80
          DO 70 J=1,17
            SUM1(J)=R(I,J)
70        CONTINUE
          CALL MULTI(SUM1,Z(I),M2,17,18)
          CALL ADD(M1,M2,M1,18)
          CALL VNORM(M1,18)
80    CONTINUE
      DO 90 I=19,84
        M1(I)=0
90    CONTINUE
C
C    ADJUST THE MP MANTISSA FOR THE EXPONENT OF X
C
      IF (EXPON)100,300,200
100     EXPON=-EXPON
        DO 120 K=1,EXPON
          IDIG=17+K
          CALL DIVI(M1,16,M2,K+16,IDIG)
          DO 110 J=1,IDIG
            M1(J)=M2(J)
110       CONTINUE
120     CONTINUE
        CALL VNORM(M1,IDIG)
        GOTO 300
200     DO 220 K=1,EXPON
          IDIG=17+K
          CALL MULTI(M1,16,M2,K+16,IDIG)
          DO 210 J=1,IDIG
            M1(J)=M2(J)
210       CONTINUE
220     CONTINUE
        CALL VNORM(M1,IDIG)
300   N=MIN0(P,84)
      DO 310 I=2,N
      Y(I)=M1(I)
310   CONTINUE
      IF (P.LE.N) RETURN
      DO 320 I=85,P
        Y(I)=0
320   CONTINUE
      RETURN
      END
      SUBROUTINE CONVI(X,II,P)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     THE SUBROUTINE CONVI(X,II,P) CONVERTS THE INTEGER II TO A        C
C     P-2 DIGIT MP NUMBER. THE MP REPRESENTATION OF II IS RETURNED     C
C     IN THE MP NUMBER X. IT IS ASSUMED THAT THE ABSOLUTE VALUE OF     C
C     II IS MACHINE REPRESENTABLE.                                     C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION X(P),Z(9)
      IX=II
      K=9
C     FIND THE SIGN OF X
      X(1)=ISIGN(1,II)
      IF (II.NE.0) GOTO 5
        X(1)=0
        RETURN
5     IX=IX*X(1)
      IF (IX.LT.10000) GOTO 40
C
C     FIND THE MANTISSA AND EXPONENT OF X
C
10      IX2=IX/10000
        Z(K)=IX-IX2*10000
        K=K-1
        IX=IX2
        IF (IX.GE.10000) GOTO 10
          Z(K)=IX
          X(2)=9-K+1
      I=3
      DO 20 J=K,9
        X(I)=Z(J)
        I=I+1
        IF (I.GT.P) RETURN
20    CONTINUE
C
C     FILL THE REMAINING DIGITS OF X WITH ZEROS IF NEEDED.
C
25    DO 30 J=I,P
        X(J)=0
30    CONTINUE
      RETURN
40    X(3)=IX
      X(2)=1
      I=4
      GOTO 25
      END
      SUBROUTINE DIV(X,Y,Z,W,P,D)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C   THE SUBROUTINE DIV(X,Y,Z,W,P,D) DIVIDES THE MP NUMBER X BY THE     C
C   MP NUMBER Y. THE RESULT IS RETURNED IN THE MP NUMBER Z. THE        C
C   DIVISION IS PERFORMED BY MULTIPLING X BY THE RECIPROCAL OF Y.      C
C   FOUR GAURD DIGITS ARE USED TO PREVENT ROUND OFF ERROR.             C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      INTEGER X,Y,Z,P,W,D
      DIMENSION X(P),Y(P),Z(P),W(D)
      K=P+4
      K1=K+1
      K2=K1+K
      CALL ASSIGN(W(1),Y,P)
      W(K)=0
      W(K-1)=0
      W(K-2)=0
      W(K-3)=0
      CALL RECIPT(W(1),W(K1),W(K2),K,D-K2)
      CALL ASSIGN(W(1),X,P)
      W(K)=0
      W(K-1)=0
      W(K-2)=0
      W(K-3)=0
      CALL MULT(W(1),W(K1),W(1),W(K2),K,2*K)
      CALL ASSIGN(Z,W(1),P)
      RETURN
      END
      SUBROUTINE DIVI(X,I,Y,P,Q)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C                                                                       C
C    THE SUBROUTINE DIVI(X,I,Y,P,Q) DIVIDES THE P-2 EXTENDED DIGIT      C
C    NUMBER X BY THE INTEGER I AND RETURNS A Q-2 EXTENDED DIGIT         C
C    RESULT IN Y.                                                       C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION X(P),Y(Q)
      CARRYD=0
C
C    FIND THE MINIMUM NUMBER OF EXTENDED DIGITS IN Y OR X.
C
      L=MIN0(P,Q)
C
C    DIVIDE X BY I SO THAT THE MINIMUM NUMBER OF EXTENDED DIGITS
C    ARE FOUND. STORE THE RESULT IN Y IN NON-NORMALIZED FORM.
C
      DO 10 J=3,L
        ITOT=(CARRYD+X(J))/I
        CARRYD=((CARRYD+X(J))-ITOT*I)*10000
        Y(J)=ITOT
 10   CONTINUE
C
C    FIND THE SIGN AND EXPONENT OF Y.
C
      Y(1)=X(1)*ISIGN(1,I)
      Y(2)=X(2)
C
C    NORMALIZE Y.
C
      CALL VNORM(Y,L)
      SHIFT=X(2)-Y(2)
C
C    IF  Y WAS IN NORMALIZED FORM AND Q WAS LESS THAN OR EQUAL TO P
C    THEN RETURN.
C
      IF(SHIFT.EQ.0.AND.Q.EQ.L) RETURN
```

```
C
C    FIND THE EXTENDED DIGITS LOST WHEN Y WAS NORMALIZED.
C    SINCE Q<P USE THE DIGITS OF X TO FIND THE DIGITS OF Y.
C
      K=L-SHIFT+1
      IF (Q.GE.P) GOTO 17
      DO 15 J=K,L
        ITOT=(CARRYD+X(L+K-J+1))/I
        CARRYD=(CARRYD+X(L+K-J+1)-ITOT*I)*10000
        Y(J)=ITOT
 15   CONTINUE
      K=L+1
      IF (K.GE.Q) RETURN
C
C    Q>P SO GENERATE THE REMAINING DIGITS OF Y FROM THE
C    REMAINDER.
C
 17   DO 20 J=K,Q
        ITOT=CARRYD/I
        CARRYD=(CARRYD-ITOT*I)*10000
        Y(J)=ITOT
 20   CONTINUE
      RETURN
      END
      SUBROUTINE EADD(X,Y,Z,P)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C     THE SUBROUTINE EADD(X,Y,Z,P) SUBTRACTS TWO MP NUMBERS WITH       C
C     EQUAL EXPONENTS. THE MP NUMBER X IS LARGER IN ABSOLUTE VALUE     C
C     THAN THE MP NUMBER Y. THE RESULT IS RETURNED IN THE MP NUMBER    C
C     Z.                                                               C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION X(P),Y(P),Z(P)
      IBRW=0
      P3=P-2
      POS=P+1
C
C     LOOP SUBTRACTING DIGITS AND GENERATING BORROWS.
C
      DO 10 J=1,P3
        POS1=POS-J
        ITOT=X(POS1)-Y(POS1)-IBRW
        IF (ITOT.GE.0) GO TO 5
          IBRW=1
          Z(POS1)=ITOT+10000
          GOTO 10
5       IBRW=0
        Z(POS1)=ITOT
10    CONTINUE
C
C     ASSIGN THE CORRECT SIGN AND EXPONENT, THEN NORMALIZE THE
C     MP RESULT.
C
      Z(2)=X(2)
      Z(1)=X(1)
      CALL VNORM(Z,P)
      RETURN
      END
      SUBROUTINE LGAM(X,A,Y,W,P,MAX)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C                                                                       C
C    THE SUBROUTINE LGAM(X,A,Y,W,P,MAX) COMPUTES THE NATURAL LOG OF     C
C    THE GAMMA FUNCTION EVALUATED AT THE MP NUMBER X. THE RESULT IS     C
C    RETURNED IN THE MP NUMBER Y. THE VARIABLE A IS A DOUBLE            C
C    PRECISION NUMBER CONTAINING THE VALUE IN X. W IS A WORK ARRAY.     C
C    LET G() BE THE GAMMA FUNCTION. THEN                                C
C                                                                       C
C       LN(G(X))=(X-1/2)*LN(X) - X + (1/2)*LN(2*PI) PLUS THE            C
C                INFINITE SUM, WHICH HAS TERMS OF THE FORM              C
C                    B(2N)/((2N-1)*(2N)*(X**(2N-1)))                    C
C                WHERE PI=3.14... AND B(2N) ARE THE BERNOULLI           C
C                NUMBERS.                                               C
C    THE ABSOLUTE ERROR IN LN(G(X)) IS LESS THAN THE FIRST TERM IN      C
C    THE INFINITE SUM THAT IS NOT USED. TO DECREASE COMPUTATION         C
C    TIME THE BERNOULLI NUMBERS UP TO B(58) HAVE BEEN STORED IN A       C
C    FORM THAT ALLOWS THEM TO BE COMPUTED TO SUFFICIENT ACCURACY.       C
C    HENCE IF B(60)/((59)*(60)*(X**59)) IS NOT SUFFICIENTLY SMALL,      C
C    I IS FOUND SUCH THAT B(60)/((59)*(60)*((X+I)**59)) IS              C
C    SMALL ENOUGH. LN(G(X+I)) IS EVALUATED AND LN(G(X)) IS FOUND AS     C
C                                                                       C
C         LN(G(X))=LN(G(X+I)) - LN((X)*(X+1)*...*(X+I-1))               C
C                                                                       C
C    THE VARIABLE A IS USED TO FIND THE VALUE OF I IF NEED BE, OR       C
C    IS USED TO COMPUTE THE FEWEST NUMBER OF TERMS NEEDED IN THE        C
C    SUMMATION TO INSURE DESIRED ACCURACY.                              C
C                                                                       C
C    THE BERNOULLI NUMBER B(2N) IS FOUND USING THE MP NUMBERS           C
C    STORED IN THE ARRAY B(K,L) AND THE INTEGER VALUES STORED           C
C    IN THE ARRAY D(K,J).                                               C
C        B(2N)=B(N,*)/(D(N,1)*D(N,2)) IF D(N,2)<>0 OTHERWISE            C
C        B(2N)=B(N,*)/(D(N,1)) .                                        C
C                                                                       C
C    THE ARRAY E(N) CONTAINS THE EXPONENT OF THE NUMBER B(2N)           C
C    AND IS USED TO DETERMINE THE NUMBER OF TERMS OF THE SUM            C
C    THAT ARE NEEDED.                                                   C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      REAL*8 A,A1,BOUND,LOGA,AT,AT2
      DIMENSION X(P),Y(P),W(MAX),B(29,11),D(29,2),E(29),LN2PI(34)
```

```
C     INITIALIZE THE ARRAY LN2PI TO THE VALUE (1/2)*LN(2*PI) FOR
C     128 DECIMAL DIGITS.
C
          LN2PI(1)=1
          LN2PI(2)=0
          LN2PI(3)=9189
          LN2PI(4)=3853
          LN2PI(5)=3204
          LN2PI(6)=6727
          LN2PI(7)=4178
          LN2PI(8)=329
          LN2PI(9)=7364
          LN2PI(10)=561
          LN2PI(11)=7639
          LN2PI(12)=8613
          LN2PI(13)=9747
          LN2PI(14)=3637
          LN2PI(15)=7834
          LN2PI(16)=5404
          LN2PI(17)=8276
          LN2PI(18)=5695
          LN2PI(19)=9272
          LN2PI(20)=6039
          LN2PI(21)=7694
          LN2PI(22)=7432
          LN2PI(23)=9863
          LN2PI(24)=5954
          LN2PI(25)=1976
          LN2PI(26)=2200
          LN2PI(27)=5583
          LN2PI(28)=6246
          LN2PI(29)=3433
          LN2PI(30)=7446
          LN2PI(31)=3668
          LN2PI(32)=5000
          LN2PI(33)=0
          LN2PI(34)=9100
       DO 5 I=1,29
         D(I,2)=0
         DO 4 J=4,11
           B(I,J)=0
4        CONTINUE
5      CONTINUE
C
C     INITIALIZE THE ARRAYS CONTAINING THE VALUES NEEDED TO
C     COMPUTE THE BERNOULLI NUMBERS AND THE ARRAY E USED TO
C     FIND THE NEEDED NUMBER OF TERMS IN THE SUM.
C
          B(1,1)=1
```

```
B(1,2)=1
B(1,3)=1
D(1,1)=6
E(1)=-1
B(2,1)=-1
B(2,2)=1
B(2,3)=1
D(2,1)=30
E(2)=-2
B(3,1)=1
B(3,2)=1
B(3,3)=1
D(3,1)=42
E(3)=-2
B(4,1)=-1
B(4,2)=1
B(4,3)=1
D(4,1)=30
E(4)=-2
B(5,1)=1
B(5,2)=1
B(5,3)=5
D(5,1)=66
E(5)=-2
B(6,1)=-1
B(6,2)=1
B(6,3)=691
D(6,1)=2730
E(6)=-1
B(7,1)=1
B(7,2)=1
B(7,3)=7
D(7,1)=6
E(7)=0
B(8,1)=-1
B(8,2)=1
B(8,3)=3617
D(8,1)=510
E(8)=0
B(9,1)=1
B(9,2)=2
B(9,3)=4
B(9,4)=3867
D(9,1)=798
E(9)=1
B(10,1)=-1
B(10,2)=2
B(10,3)=17
B(10,4)=4611
D(10,1)=330
E(10)=2
```

```
B(11,1)=1
B(11,2)=2
B(11,3)=85
B(11,4)=4513
D(11,1)=138
E(11)=3
B(12,1)=-1
B(12,2)=3
B(12,3)=2
B(12,4)=3636
B(12,5)=4091
D(12,1)=2730
E(12)=4
B(13,1)=1
B(13,2)=2
B(13,3)=855
B(13,4)=3103
D(13,1)=6
E(13)=6
B(14,1)=-1
B(14,2)=3
B(14,3)=237
B(14,4)=4946
B(14,5)=1029
D(14,1)=870
E(14)=7
B(15,1)=1
B(15,2)=4
B(15,3)=8
B(15,4)=6158
B(15,5)=4127
B(15,6)=6005
D(15,1)=7161
D(15,2)=2
E(15)=8
B(16,1)=-1
B(16,2)=4
B(16,3)=7
B(16,4)=7093
B(16,5)=2104
B(16,6)=1217
D(16,1)=510
E(16)=10
B(17,1)=1
B(17,2)=4
B(17,3)=2
B(17,4)=5776
B(17,5)=8785
B(17,6)=8367
D(17,1)=6
```

```
E(17)=11

B(18,1)=-1
B(18,2)=5
B(18,3)=2631
B(18,4)=5271
B(18,5)=5530
B(18,6)=5347
B(18,7)=7373
D(18,1)=10101
D(18,2)=190
E(18)=13
B(19,1)=1
B(19,2)=4
B(19,3)=2929
B(19,4)=9939
B(19,5)=1384
B(19,6)=1559
D(19,1)=6
E(19)=14
B(20,1)=-1
B(20,2)=6
B(20,3)=2
B(20,4)=6108
B(20,5)=2718
B(20,6)=4964
B(20,7)=4912
B(20,8)=2051
D(20,1)=1353
D(20,2)=10
E(20)=16
B(21,1)=1
B(21,2)=6
B(21,3)=15
B(21,4)=2009
B(21,5)=7643
B(21,6)=9180
B(21,7)=7080
B(21,8)=2691
D(21,1)=1806
E(21)=17
B(22,1)=-1
B(22,2)=6
B(22,3)=278
B(22,4)=3326
B(22,5)=9579
B(22,6)=3010
B(22,7)=2423
B(22,8)=5023
D(22,1)=690
```

```
E(22)=19
B(23,1)=1
B(23,2)=6
B(23,3)=5964
B(23,4)=5111
B(23,5)=1593
B(23,6)=9121
B(23,7)=6327
B(23,8)=7961
D(23,1)=282
E(23)=21
B(24,1)=-1
B(24,2)=7
B(24,3)=5609
B(24,4)=4033
B(24,5)=6899
B(24,6)=7817
B(24,7)=6862
B(24,8)=4912
B(24,9)=7547
D(24,1)=4641
D(24,2)=10
E(24)=23
B(25,1)=1
B(25,2)=7
B(25,3)=495
B(25,4)=572
B(25,5)=542
B(25,6)=1079
B(25,7)=6482
B(25,8)=1247
B(25,9)=7547
D(25,1)=66
E(25)=24
B(26,1)=-1
B(26,2)=8
B(26,3)=80
B(26,4)=1165
B(26,5)=7181
B(26,6)=3548
B(26,7)=1853
B(26,8)=3479
B(26,9)=2499
B(26,10)=1853
D(26,1)=1590
E(26)=26
B(27,1)=1
B(27,2)=8
B(27,3)=2914
B(27,4)=9963
```

```
          B(27,5)=6348
          B(27,6)=8486
          B(27,7)=2421
          B(27,8)=4181
          B(27,9)=2381
          B(27,10)=2691
          D(27,1)=798
          E(27)=28
          B(28,1)=-1
          B(28,2)=9
          B(28,3)=24
          B(28,4)=7939
          B(28,5)=2929
          B(28,6)=3132
          B(28,7)=2675
          B(28,8)=3685
          B(28,9)=4157
          B(28,10)=3966
          B(28,11)=3229
          D(28,1)=870
          E(28)=30
          B(29,1)=1
          B(29,2)=9
          B(29,3)=844
          B(29,4)=8361
          B(29,5)=3348
          B(29,6)=8800
          B(29,7)=4168
          B(29,8)=2046
          B(29,9)=7759
          B(29,10)=9403
          B(29,11)=6021
          D(29,1)=354
          E(29)=32
      I=0
C
C    FIND IF X IS LARGE ENOUGH THAT 29 TERMS SUPPLY SUFFICIENT
C    ACCURACY.
C
C
C    X IS TOO SMALL FOR 29 TERMS TO SUPPLY SUFFICIENT ACCURACY.
C    FIND I SUCH THAT LN(G(X+I)) CAN BE EVALUATED USING ONLY
C    29 TERMS OF THE SUM.
C
      BOUND=(DFLOAT(-4*(P-1)-32)+DLOG10(DFLOAT(60*59)))/59.D0
      LOGA=-DLOG10(A)
      IF (LOGA.LT.BOUND) GOTO  120
        A1=A
        RTERMS=29
10      A1=A1+1.D0
```

```
        I=I+1
        IF (-DLOG10(A1).LT.BOUND) GOTO 20
        GOTO 10
20    DECN=P+3
      DECN2=DECN*2
      CDIV=DECN+1
      WORK=CDIV+DECN
      CALL ASSIGN(W(1),X,P)
      W(P+1)=0
      W(P+2)=0
      W(P+3)=0
      W(CDIV)=1
      W(CDIV+1)=1
      W(CDIV+2)=1
      J=DECN-1
      DO 25 K=3,J
        W(CDIV+K)=0
25    CONTINUE
C
C     FIND THE PRODUCT (X*(X+1)*....*(X+I-1)) AND STORE IN
C     W(CDIV). ALSO FIND THE MP VALUE X+I AND STORE IN W(1)
C
      DO 30 J=1,I
        CALL MULT(W(1),W(CDIV),W(CDIV),W(WORK),DECN,DECN2)
        CALL ADD1(W(1),DECN)
30    CONTINUE
      GOTO 50
40    CALL ASSIGN(W(1),X,P)
      W(P+1)=0
      W(P+2)=0
50    ND=P+2
      LNX=2*ND+3
      ND2=ND*2
      XM12=LNX+ND
      WORK=XM12+ND
C
C     LET X DENOTE X OR X+I AS APPROPRIATE
C
C     FIND THE VALUE (X-1/2)*LN(X)-X+(1/2)*LN(2*PI) AND STORE IN
C     W(LNX).
C
      CALL LN(W(1),W(LNX),ND,W(XM12),21*ND+288+21*(ND/50))
      CALL CONVI(W(XM12),1,ND)
      CALL DIVI(W(XM12),2,W(WORK),ND,ND)
      W(WORK)=-1
      CALL ADD(W(1),W(WORK),W(XM12),ND)
      CALL MULT(W(XM12),W(LNX),W(LNX),W(WORK),ND,ND2)
      W(1)=-1
      CALL ADD(W(1),W(LNX),W(LNX),ND)
      CALL ADD(W(LNX),LN2PI,W(LNX),ND)
      W(1)=1
      E(1)=XM12
C
C   SET UP POINTERS FOR THE MP NUMBERS THAT CONTAIN THE VALUES
C   B(2N)/((2N-1)*(2N)*(X**(2N-1)), SO THAT THE TERMS CAN BE
C   SUMMED FROM SMALLEST TO LARGEST.
C
C
C   ZPOW IS A POINTER TO THE VALUE 1/(X**(2N-1)) AND Z2 IS A
C   POINTER TO THE VALUE 1/(X**2).
C
      DO 60 J=2,RTERMS
        E(J)=E(J-1)+ND
60    CONTINUE
      ZPOW=30*ND+XM12
      Z2=ZPOW+ND
      WORK=Z2+ND
      CALL RECIPT(W(1),W(1),W(XM12),ND,6*ND+24)
      CALL MULT(W(1),W(1),W(Z2),W(WORK),ND,ND2)
      DO 70 J=1,11
        W(XM12+J-1)=B(1,J)
70    CONTINUE
C
C   FIND B(2) AND THE FIRST TERM OF THE SUMATION.
C
      CALL DIVI(W(XM12),D(1,1),W(WORK),11,ND)
      CALL DIVI(W(WORK),2,W(XM12),ND,ND)
      CALL MULT(W(XM12),W(1),W(XM12),W(WORK),ND,ND2)
      CALL ASSIGN(W(ZPOW),W(1),ND)
C
C   IF MORE THAN 1 TERM OF THE SUM IS NEEDED, GO TO COMPUTE
C   THE REQUIRED NUMBER OF TERMS.
C
      IF (RTERMS.GT.1) GOTO 75
C
C   ONLY 1 TERM OF THE SUM WAS REQUIRED. REASSEMBLE THE
C   VALUE OF LN(G(X)) AND RETURN.
C
      CALL ADD(W(XM12),W(LNX),W(1),ND)
      CALL ASSIGN(Y,W(1),P)
      RETURN
C
C    FIND VALUES OF THE NEEDED TERMS IN THE SUM.
C
75    DO 90 J=2,RTERMS
        CALL MULT(W(ZPOW),W(Z2),W(ZPOW),W(WORK),ND,ND2)
        PLACE=E(J)-1
        DO 80 K=1,11
          W(PLACE+K)=B(J,K)
80      CONTINUE
```

```
         PLACE=PLACE+1
C
C     FIND B(2N)/((2N-1)*(2N)*(X**(2N-1)))
C
         CALL DIVI(W(PLACE),D(J,1),W(WORK),11,ND)
         CALL DIVI(W(WORK),(2*J)*(2*J-1),W(PLACE),ND,ND)
         IF (D(J,2).EQ.0) GOTO 85
           CALL DIVI(W(PLACE),D(J,2),W(WORK),ND,ND)
           CALL ASSIGN(W(PLACE),W(WORK),ND)
85       CALL MULT(W(PLACE),W(ZPOW),W(PLACE),W(WORK),ND,ND2)
90     CONTINUE
       CALL ASSIGN(W(1),W(PLACE),ND)
       BACK=RTERMS
       RTERMS=RTERMS-1
C
C     SUM THE TERMS B(2N)/((2N-1)*(2N)*(X**(2N-1))) FROM
C     SMALLEST TO LARGEST.
C
       DO 100 J=1,RTERMS
         CALL ADD(W(1),W(E(BACK-J)),W(1),ND)
100    CONTINUE
C
C     REASSEMBLE THE VALUE OF LN(G(X))
C
       CALL ADD(W(1),W(LNX),W(1),ND)
       IF (I.EQ.0) GOTO 110
C
C     THE VALUE LN(G(X+I)) WAS FOUND. NOW FIND LN(G(X))
C
         CALL LN(W(CDIV),W(CDIV),ND,W(XM12),21*ND+288+21*(ND/50))
         W(CDIV)=-1
         CALL ADD(W(1),W(CDIV),W(1),ND)
C
C     ASSIGN THE VALUE OF LN(G(X)) TO THE MP NUMBER Y AND RETURN
C
110    CALL ASSIGN(Y,W(1),P)
       RETURN
C
C     FIND THE FEWEST NUMBER OF TERMS NEEDED TO EVALUATE LN(G(X))
C     TO THE DESIRED ACCURRACY.
C
120    TOP=29
       BOTTOM=1
       IF(LOGA.GE.DFLOAT(-4*(P-1)+1)+DLOG10(2.D0)) GOTO 130
       RTERMS=1
       GOTO 40
130    TRY=(TOP+BOTTOM)/2
       BOUND=(DFLOAT(-4*(P-1)-E(TRY))+DLOG10(DFLOAT((2*TRY)*
     1 (2*TRY-1))))/DFLOAT(2*TRY-1)
       IF (LOGA.LT.BOUND) GOTO 140
```

```
      BOTTOM=TRY
      GOTO 150
140   TOP=TRY
150   IF (TOP-BOTTOM.NE.1) GOTO 130
      RTERMS=TOP
      GOTO 40
      END
      SUBROUTINE LN(X,Y,P,W,MAX)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C    THE SUBROUTINE LN(X,Y,P,W,MAX) EVALUATES THE NATURAL LOG OF       C
C    THE MP NUMBER X AND RETURNS THE RESULT IN THE MP NUMBER Y.        C
C    W IS A WORK ARRAY. THE METHOD USED TO EVALUATE LN(X) IS AS        C
C    FOLLOWS.                                                          C
C       WRITE    X=(10**N)*F      WHERE .1=<F<1 .                      C
C       LET      F'=10*F .                                             C
C       FIND K  SUCH THAT 1=<F'/(2**K)<2 .                             C
C       LET      F''=F'/(2**K)   .                                     C
C       LN(X)=LN(10)*(N-1) + LN(F'') + K*LN(2).                        C
C                                                                      C
C    TO FIND LN(F'') A CONTINUED FRACTION IS USED. THE                 C
C    DIFFERENCE BETWEEN CONVERGENTS BOUND THE ERROR IN THE             C
C    EVALUATION OF LN(F''). LET R=F''-1. THEN                          C
C    LN(F'')=(R/(1+R/(2+R/(3+2R/(2+2R/(5+3R/(2+3R/(7+4R/(2+......      C
C                                                                      C
C    THE GENERAL TERMS ARE OF THE FORM ((N/2)*R)/2 FOR N EVEN          C
C    AND ((N-1)/2*R)/N FOR N ODD, WHEN N>3.                            C
C                                                                      C
C                                                                      C
C    TO FIND THE NTH CONVERGENT TWO QUANITIES P(N) AND Q(N) ARE
C    NEEDED. THE NTH CONVEGENRGENT V(N)=P(N)/Q(N). FOR N>1 P(N)
C    AND Q(N) ARE DEFINED AS FOLLOWS.
C      P(N)=N*P(N-1) + (((N-1)/2)*R)*P(N-2) FOR N ODD
C      P(N)=2*P(N-1) + ((N/2)*R)*P(N-2)  FOR N EVEN
C      Q(N)=N*Q(N-1) + (((N-1)/2)*R)*P(N-2) FOR N ODD
C      Q(N)=2*Q(N-1) + ((N/2)*R)*P(N-2)  FOR N EVEN
C
C      P(0)=0
C      P(1)=R
C      Q(0)=1
C      Q(1)=1
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION X(P),Y(P),W(MAX),LN2(40),LN10(40)
C
C    INITIALIZE THE VALUE OF LN(2) TO 152 DECIMAL DIGITS
C
         LN2(1)=1
         LN2(2)=0
         LN2(3)=6931
         LN2(4)=4718
         LN2(5)=559
         LN2(6)=9453
         LN2(7)=941
      LN2(8)=7232
      LN2(9)=1214
      LN2(10)=5817
      LN2(11)=6568
      LN2(12)=755
      LN2(13)=13
      LN2(14)=4360
      LN2(15)=2552
      LN2(16)=5412
      LN2(17)=680
      LN2(18)=94
      LN2(19)=9339
      LN2(20)=3621
      LN2(21)=9696
      LN2(22)=9471
      LN2(23)=5605
      LN2(24)=8633
      LN2(25)=2699
      LN2(26)=6418
      LN2(27)=6875
      LN2(28)=4200
      LN2(29)=1481
      LN2(30)=0205
      LN2(31)=7068
      LN2(32)=5733
      LN2(33)=6855
      LN2(34)=2023
      LN2(35)=5758
      LN2(36)=1305
      LN2(37)=5703
      LN2(38)=2670
      LN2(39)=7516
      LN2(40)=3507
C
C    INITIALIZE THE VAULE OF LN(10) TO 149 DECIMAL DIGITS
C
      LN10(1)=1
      LN10(2)=1
      LN10(3)=2
      LN10(4)=3025
      LN10(5)=8509
      LN10(6)=2994
      LN10(7)=456
      LN10(8)=8401
      LN10(9)=7991
      LN10(10)=4546
      LN10(11)=8436
      LN10(12)=4207
      LN10(13)=6011
      LN10(14)=148
```

```
          LN10(15)=8628
          LN10(16)=7729
          LN10(17)=7603
          LN10(18)=3327
          LN10(19)=9009
          LN10(20)=6757
          LN10(21)=2609
          LN10(22)=6773
          LN10(23)=5248
          LN10(24)=235
          LN10(25)=9972
          LN10(26)=508
          LN10(27)=9598
          LN10(28)=2983
          LN10(29)=4196
          LN10(30)=7595
          LN10(31)=422
          LN10(32)=8624
          LN10(33)=8633
          LN10(34)=4095
          LN10(35)=2546
          LN10(36)=5082
          LN10(37)=8067
          LN10(38)=5666
          LN10(39)=6287
          LN10(40)=3690
C
C    FIND F' AND ASSIGN TO W(1). FIGURE ADJUSTMENT NEEDED FOR THE
C    EXPONENT.
C
      IF (X(3).LT.1000) GOTO 5
        ADJ=1
        CALL DIVI(X,1000,W(1),P,P+1)
        GOTO 20
5     IF (X(3).LT.100) GOTO 10
        ADJ=2
        CALL DIVI(X,100,W(1),P,P+1)
        GOTO 20
10    IF (X(3).LT.10) GOTO 15
        ADJ=3
        CALL DIVI(X,10,W(1),P,P+1)
        GOTO 20
15      ADJ=4
        W(P+1)=0
        CALL ASSIGN(W(1),X,P)
20    W(2)=1
      W(P+2)=0
      ND=P+2
C
```

```
C    FIND N-1 AND STORE IN POWER.
C
      POWER=4*X(2)-ADJ
      I=0
30    IF (W(3).LE.2) GOTO 40
C
C    REDUCE W(1)=F' TO F'' AND STORE F'' IN W(1).
C
35          CALL DIVI(W(1),2,W(ND+1),ND,ND+1)
        I=I+1
        CALL ASSIGN(W(1),W(ND+1),ND)
        GOTO 30
40    CALL ASSIGN(W(ND+1),W(1),ND)
C
C    CHECK THAT 1=<F''<2 . IF F''>2 BRANCH TO REDUCE F''.
C
      W(ND+1)=-1
      CALL ADD1(W(ND+1),ND)
      CALL ADD1(W(ND+1),ND)
      IF (W(ND+1).LT.0) GOTO 35
C
C    FIND R=F''-1 AND STORE IN W(1)
C
      W(1)=-1
      CALL ADD1(W(1),ND)
      W(1)=-1*W(1)
C
C    SREQ IS THE SPACE REQUIRED FOR EACH MP NUMBER. NOTE TWO
C    DIGITS HAVE BEEN ADDED TO GAURD AGAINST ROUND OFF ERROR.
C
      SREQ=ND+2
C
C     POINTER        DESCRIPTION
C       P1              P(N-2)
C       P2              P(N-1)
C       P3              P(N)
C       Q1              Q(N-2)
C       Q2              Q(N-1)
C       Q3              Q(N)
C       B               (N/2)*R FOR N EVEN
C                       ((N-1)/2)*R FOR N ODD
C       V1              V(N-1)
C       V2              V(N)
C       I1              INTERMIDIATE RESULT
C       I2              INTERMIDIATE RESULT AND WORK SPACE
C
      P1=SREQ+1
      P2=P1+SREQ
      P3=P2+SREQ
      Q1=P3+SREQ
      Q2=Q1+SREQ
      Q3=Q2+SREQ
      B=Q3+SREQ
      V1=B+SREQ
      V2=V1+SREQ
      I1=V2+SREQ
C
C     IF R IS ZERO BRANCH TO REASSEMBLE LN(X)
C
      I2=I1+SREQ+2
      IF (W(1).EQ.0) GOTO 85
      CALL CONVI(W(Q1),1,SREQ)
      SRQ2=2*SREQ
      ND1=ND+1
      DO 50 K=ND1,SREQ
        W(K)=0
50    CONTINUE
C
C     FIND THE VALUES P(1),P(2),Q(1), AND Q(2)
C
      CALL ASSIGN(W(B),W(1),SREQ)
      CALL ASSIGN(W(Q2),W(B),SREQ)
      CALL ASSIGN(W(P1),W(B),SREQ)
      CALL ADD(W(P1),W(P1),W(P2),SREQ)
      CALL ADD1(W(Q2),SREQ)
      CALL ADD1(W(Q2),SREQ)
      SREQ2=SREQ+1
C
C     FIND THE THIRD CONVERGENT.
C
      CALL MULT(W(P1),W(B),W(I1),W(I2),SREQ,SRQ2)
      CALL MULTI(W(P2),3,W(I2),SREQ,SREQ2)
      CALL ADD(W(I1),W(I2),W(P3),SREQ)
      CALL MULT(W(Q1),W(B),W(I1),W(I2),SREQ,SRQ2)
      CALL MULTI(W(Q2),3,W(I2),SREQ,SREQ2)
      CALL ADD(W(I1),W(I2),W(Q3),SREQ)
      CALL DIV(W(P3),W(Q3),W(V1),W(I1),SREQ,8*SREQ+80)
C
C     RESET POINTERS FOR NEXT CONVERGENT
C
      C=P1
      P1=P2
      P2=P3
      P3=C
      C=Q1
      Q1=Q2
      Q2=Q3
      Q3=C
      A=3
```

```
C
C     FIND THE VALUE OF EVEN CONVERGENTS
C
60      CALL ADD(W(B),W(1),W(B),SREQ)
        CALL MULT(W(P1),W(B),W(I1),W(I2),SREQ,SRQ2)
        CALL MULTI(W(P2),2,W(I2),SREQ,SREQ2)
        CALL ADD(W(I1),W(I2),W(P3),SREQ)
        CALL MULT(W(Q1),W(B),W(I1),W(I2),SREQ,SRQ2)
        CALL MULTI(W(Q2),2,W(I2),SREQ,SREQ2)
        CALL ADD(W(I1),W(I2),W(Q3),SREQ)
        CALL DIV(W(P3),W(Q3),W(V2),W(I1),SREQ,8*SREQ+80)
C
C     CHECK FOR ACHIEVEMENT OF DESIRED ACCURACY
C
        DO 65 K=1,P
          IF(W(V2-1+K)-W(V1-1+K)) 67,65,67
65      CONTINUE
        GOTO 80
C
C     RESET POINTERS FOR THE NEXT CONVERGENT
C
67      C=P1
        P1=P2
        P2=P3
        P3=C
        C=Q1
        Q1=Q2
        Q2=Q3
        Q3=C
        C=V1
        V1=V2
        V2=C
        A=A+2
C
C    FIND THE ODD CONVERGENTS
C
        CALL MULT(W(P1),W(B),W(I1),W(I2),SREQ,SRQ2)
        CALL MULTI(W(P2),A,W(I2),SREQ,SREQ2)
        CALL ADD(W(I1),W(I2),W(P3),SREQ)
        CALL MULT(W(Q1),W(B),W(I1),W(I2),SREQ,SRQ2)
        CALL MULTI(W(Q2),A,W(I2),SREQ,SREQ2)
        CALL ADD(W(I1),W(I2),W(Q3),SREQ)
        CALL DIV(W(P3),W(Q3),W(V2),W(I1),SREQ,8*SREQ+80)
C
C    RESET THE POINTERS FOR THE NEXT CONVERGENT
C
        C=P1
        P1=P2
        P2=P3
        P3=C
```

```
      C=Q1
      Q1=Q2
      Q2=Q3
      Q3=C
      C=V1
      V1=V2
      V2=C
C
C    CHECK FOR ACHIEVEMENT OF DESIRED ACCURACY
C
      DO 70 K=1,P
        J=K-1
        IF(W(V2+J)-W(V1+J)) 60,70,60
70    CONTINUE
C
C    REASSEMBLE THE VALUE OF LN(X)
C
80    IF (I.EQ.0) GOTO 90
        CALL ADD(W(V2),LN2,W(V1),SREQ)
        CALL ASSIGN(W(V2),W(V1),SREQ)
        I=I-1
        GOTO 80
85    W(V1)=0
90    CALL MULTI(LN10,POWER,W(1),ND,ND+1)
      CALL ADD(W(1),W(V1),W(V2),ND)
      CALL ASSIGN(Y,W(V2),P)
      RETURN
      END
      SUBROUTINE MEXP(X,Y,P,W,MAX)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C    THE SUBROUTINE MEXP(X,Y,P,W,MAX) RETURNS THE EXPONENTIAL          C
C    FUNCTION EVALUATED AT THE MP NUMBER X IN THE MP NUMBER Y. THE     C
C    VALUE EXP(X) IS EVALUATED AS FOLLOWS.                             C
C                                                                      C
C       FIND N AND G SUCH THAT X=N*LN(10)+G                            C
C          WHERE N IS AN INTEGER AND ABSOLUTE VALUE OF G=<LN(10)/2 .   C
C                                                                      C
C       THEN EXP(X)=EXP(G)*10**N .                                     C
C                                                                      C
C       FIND R AND K WITH ABSOLUTE VALUE OF R<.0008                    C
C       SUCH THAT R=G/(2**K) .                                         C
C                                                                      C
C       EXP(X)=(EXP(R)**(2**K))*10**N                                  C
C                                                                      C
C       TO EVALUATE EXP(R) A CONTINUED FRACTION IS USED. THE           C
C       CONVERGENTS 2,3,6,7,10,11,... FORM A BOUND ON EXP(R) FROM      C
C       ONE SIDE, WHILE THE CONVERGENTS 4,5,8,9,12,13,...  FORM A      C
C       BOUND ON EXP(R) FROM THE OTHERSIDE.                            C
C                                                                      C
C       LN(R)=1+R/(1-R/(2+R/(3-R/(2+R/(5-R/(2+R/(7-R/(2+.........      C
C                                                                      C
C       THE NTH CONVERGENT IS V(N)=P(N)/Q(N).                          C
C       P(0)=0                                                         C
C       P(1)=R                                                         C
C       Q(0)=1                                                         C
C       Q(1)=1                                                         C
C                                                                      C
C       P(N)=2*P(N-1)-R*P(N-2)     FOR N EVEN                          C
C       P(N)=N*P(N-1)+R*P(N-2)     FOR N ODD                           C
C       Q(N)=2*Q(N-1)-R*Q(N-2)     FOR N EVEN                          C
C       Q(N)=N*Q(N-1)+R*Q(N-2)     FOR N ODD                           C
C                                                                      C
C    THREE EXTRA DIGITS ARE USED TO GAURD AGAINST ROUND OFF            C
C    ERROR.                                                            C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION X(P),Y(P),W(MAX),LN10(40),RLN10(40)
C
C    INITIALIZE THE VALUE LN(10) TO 149 DECIMAL DIGITS
C
         LN10(1)=1
         LN10(2)=1
      LN10(3)=2
      LN10(4)=3025
      LN10(5)=8509
      LN10(6)=2994
      LN10(7)=456
      LN10(8)=8401
      LN10(9)=7991
      LN10(10)=4546
      LN10(11)=8436
      LN10(12)=4207
      LN10(13)=6011
      LN10(14)=148
      LN10(15)=8628
      LN10(16)=7729
      LN10(17)=7603
      LN10(18)=3327
      LN10(19)=9009
      LN10(20)=6757
      LN10(21)=2609
      LN10(22)=6773
      LN10(23)=5248
      LN10(24)=235
      LN10(25)=9972
      LN10(26)=508
      LN10(27)=9598
      LN10(28)=2983
      LN10(29)=4196
      LN10(30)=7595
      LN10(31)=422
      LN10(32)=8624
      LN10(33)=8633
      LN10(34)=4095
      LN10(35)=2546
      LN10(36)=5082
      LN10(37)=8067
      LN10(38)=5666
      LN10(39)=6287
      LN10(40)=3690
C
C    INITIALIZE THE VALUE 1/LN(10) TO 152 DECIMAL DIGITS
C
      RLN10(1)=1
      RLN10(2)=0
      RLN10(3)=4342
      RLN10(4)=9448
      RLN10(5)=1903
      RLN10(6)=2518
      RLN10(7)=2765
      RLN10(8)=1128
      RLN10(9)=9189
```

```
          RLN10(10)=1660
          RLN10(11)=5082
          RLN10(12)=2943
          RLN10(13)=9700
          RLN10(14)=5803
          RLN10(15)=6665
          RLN10(16)=6611
          RLN10(17)=4453
          RLN10(18)=7831
          RLN10(19)=6586
          RLN10(20)=4649
          RLN10(21)=2088
          RLN10(22)=7077
          RLN10(23)=4729
          RLN10(24)=2249
          RLN10(25)=4933
          RLN10(26)=8431
          RLN10(27)=7483
          RLN10(28)=1870
          RLN10(29)=6142
          RLN10(30)=3923
          RLN10(31)=7703
          RLN10(32)=8928
          RLN10(33)=6743
          RLN10(34)=185
          RLN10(35)=9755
          RLN10(36)=8273
          RLN10(37)=307
          RLN10(38)=4343
          RLN10(39)=9198
          RLN10(40)=8448
      ND=P+3
      SHIFT=0
      POW=0
      N=P+1
      ND1=ND+1
C
C    IF X=0 RETURN THE VALUE 1
C
      IF (X(1).NE.0) GOTO 1
        CALL CONVI(Y,1,P)
        RETURN
C
C    DETERMINE THE VALUE N  AND STORE IN POW
C
1     CALL MULT(X,RLN10,W(1),W(P+1),P,2*P)
      D=W(2)
      IF (D) 60,30,10
10    D2=D+2
      IBASE=1
```

```
20     POW=POW+W(D2)*IBASE
       IBASE=IBASE*10000
       D2=D2-1
       IF (D2.GT.3) GOTO 20
30     IF(W(D+3).GE.5000) POW=POW+1
       POW=POW*X(1)
       IF (POW.EQ.0) GOTO 60
       CALL ASSIGN(W(1),X,P)
       DO 40 J=N,ND
         W(J)=0
40     CONTINUE
C
C     DETERMINE THE VALUE G AND STORE IN W(1)
C
       CALL MULTI(LN10,POW,W(ND1),ND,ND1)
       W(ND1)=W(ND1)*(-1)
       CALL ADD(W(1),W(ND1),W(1),ND)
C
C     FIND THE VALUE R=(G/(2**K)) SUCH THAT ABSOLUTE VALUE R<.0008
C
45     IF (W(2).LT.0) GOTO 80
         W(ND1)=0
         W(ND1+1)=0
         ND3=ND+3
         ND2=ND+2
50     IF(W(2).LT.0) GOTO 80
         IF(W(2).EQ.0.AND.W(3).LT.8) GOTO 80
           CALL DIVI(W(1),2,W(ND3),ND2,ND2)
           SHIFT=SHIFT+1
           CALL ASSIGN(W(1),W(ND3),ND2)
           GOTO 50
60     CALL ASSIGN(W(1),X,P)
       P1=P+1
       DO 70 J=P1,ND
         W(J)=0
70     CONTINUE
       GOTO 45
80     IF (SHIFT.NE.0) ND=ND2
C
C     POINTER     DESCRIPTION
C       1            R
C       P1           P(N-2)
C       P2           P(N-1)
C       P3           P(N)
C       Q1           Q(N-2)
C       Q2           Q(N-1)
C       Q3           Q(N)
C       V1           V(N-2)
C       V2           V(N-1)
C       V3           V(N)
```

```
C       I1             INTERMIDIATE RESULT 1
C       I2             INTERMIDIATE RESULT AND WORK SPACE
C
      P1=ND+1
      P2=P1+ND
      P3=P2+ND
      Q1=P3+ND
      Q2=Q1+ND
      Q3=Q2+ND
      V1=Q3+ND
      V2=V1+ND
      V3=V2+ND
      I1=V3+ND
      I2=I1+ND
C
C     FIND THE VALUES OF P(0),P(1),Q(0), AND Q(1)
C
      W(P1)=0
      CALL CONVI(W(Q1),1,ND)
      CALL ASSIGN(W(P2),W(1),ND)
      CALL ASSIGN(W(Q2),W(Q1),ND)
      ND2=ND*2
      ND1=ND+1
C
C     FIND THE SECOND CONVERGENT
C
      W(1)=-1*W(1)
      CALL MULT(W(1),W(P1),W(I1),W(I2),ND,ND2)
      CALL MULTI(W(P2),2,W(I2),ND,ND1)
      CALL ADD(W(I2),W(I1),W(P3),ND)
      CALL MULT(W(1),W(Q1),W(I1),W(I2),ND,ND2)
      CALL MULTI(W(Q2),2,W(I2),ND,ND1)
      CALL ADD(W(I1),W(I2),W(Q3),ND)
C
C     RESET THE POINTERS FOR THE NEXT CONVERGENT
C
      C=P3
      P3=P1
      P1=P2
      P2=C
      C=Q3
      Q3=Q1
      Q1=Q2
      Q2=C
C
C     FIND THE THIRD CONVERGENT
C
      W(1)=-1*W(1)
      ND8=8*ND+80
      CALL MULT(W(1),W(P1),W(I1),W(I2),ND,ND2)
      CALL MULTI(W(P2),3,W(I2),ND,ND1)
      CALL ADD(W(I1),W(I2),W(P3),ND)
      CALL MULT(W(1),W(Q1),W(I1),W(I2),ND,ND2)
      CALL MULTI(W(Q2),3,W(I2),ND,ND1)
      CALL ADD(W(I1),W(I2),W(Q3),ND)
      CALL DIV(W(P3),W(Q3),W(V3),W(I1),ND,ND8)
C
C     RESET THE POINTERS FOR THE NEXT CONVERGENT
C
      A=3
      C=P3
      P3=P1
      P1=P2
      P2=C
      C=Q3
      Q3=Q1
      Q1=Q2
      Q2=C
      C=V3
      V3=V1
      V1=V2
      V2=C
      W(1)=-1*W(1)
C
C    FIND THE VALUE OF EVEN CONVERGENTS
C
90    CALL MULT(W(1),W(P1),W(I1),W(I2),ND,ND2)
      CALL MULTI(W(P2),2,W(I2),ND,ND1)
      CALL ADD(W(I1),W(I2),W(P3),ND)
      CALL MULT(W(1),W(Q1),W(I1),W(I2),ND,ND2)
      CALL MULTI(W(Q2),2,W(I2),ND,ND2)
      CALL ADD(W(I1),W(I2),W(Q3),ND)
      CALL DIV(W(P3),W(Q3),W(V3),W(I1),ND,ND8)
C
C    CHECK IF DESIRED ACCURACY HAS BEEN ACHIEVED
C
      DO 100 J=1,P
        IF (W(V3+J-1)-W(V2+J-1)) 110,100,110
100   CONTINUE
      GOTO 140
C
C    RESET POINTERS FOR THE NEXT CONVERGENT
C
110   C=P3
      P3=P1
      P1=P2
      P2=C
      C=Q3
      Q3=Q1
      Q1=Q2
      Q2=C
      C=V3
      V3=V1
      V1=V2
      V2=C
      W(1)=-1*W(1)
      A=A+2
C
C    FIND THE VALUE OF ODD CONVERGENTS
C
      CALL  MULT(W(1),W(P1),W(I1),W(I2),ND,ND2)
      CALL MULTI(W(P2),A,W(I2),ND,ND1)
      CALL ADD(W(I1),W(I2),W(P3),ND)
      CALL MULT(W(1),W(Q1),W(I1),W(I2),ND,ND2)
      CALL MULTI(W(Q2),A,W(I2),ND,ND1)
      CALL ADD(W(I1),W(I2),W(Q3),ND)
      CALL DIV(W(P3),W(Q3),W(V3),W(I1),ND,ND8)
C
C    CHECK IF DESIRED ACCURACY ACHIEVED
C
      DO 120 J=1,P
        IF (W(V3-1+J)-W(V1+J-1)) 130,120,130
120   CONTINUE
      GOTO 140
C
C    RESET POINTERS FOR NEXT CONVERGENT
C
130   C=P3
      P3=P1
      P1=P2
      P2=C
      C=Q3
      Q3=Q1
      Q1=Q2
      Q2=C
      C=V3
      V3=V1
      V1=V2
      V2=C
      W(1)=-1*W(1)
      GOTO 90
C
C    REASSEMBLE THE VALUE OF EXP(X)
C
140   CALL ASSIGN(W(1),W(V3),ND)
       CALL ADD1(W(1),ND)
141   IF (SHIFT.EQ.0) GOTO 145
        CALL MULT(W(1),W(1),W(V2),W(I1),ND,ND2)
        CALL ASSIGN(W(1),W(V2),ND)
        SHIFT=SHIFT-1
        GOTO 141
 145   IP=POW/4
       W(2)=W(2)+IP
       N10=POW-IP*4
       IF (N10) 160,150,170
 150   CALL ASSIGN(Y,W(1),P)
       RETURN
 160    N10=INT(10.**(-N10))
       CALL DIVI(W(1),N10,W(P1),ND,ND1)
       CALL ASSIGN(Y,W(P1),P)
       RETURN
 170   CALL MULTI(W(1),INT(10.**N10),W(P1),ND,ND1)
       CALL ASSIGN(Y,W(P1),P)
       RETURN
       END
```

```
      SUBROUTINE MULLER(P,Q,X,PQ,X1,IBETA,W,PREC,MAX,PX,QX,PQX)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C    THE SUBROUTINE MULLER COMPUTES THE INCOMPLETE BETA FUNCTION       C
C    I(P,Q;X) USING MULLERS CONTINUED FRACTION. THE RESULT IS  A MP    C
C    NUMBER RETURNED IN IBETA.                                         C
C                                                                      C
C    SUBROUTINE PARAMETER   DESCRIPTION                                C
C        P                  MP NUMBER CONTAINING THE VALUE OF P        C
C                           IN I(P,Q;X)                                C
C        Q                  MP NUMBER CONTAINING THE VALUE OF Q        C
C                           IN I(P,Q;X)                                C
C        X                  MP NUMBER CONTAINING THE VALUE OF X        C
C                           IN I(P,Q;X)                                C
C        PQ                 MP NUMBER WITH THE VALUE P+Q               C
C        X1                 MP NUMBER CONTAINING 1-X                   C
C        IBETA              MP NUMBER RETURNED WITH THE VALUE          C
C                           I(P,Q;X)                                   C
C        W                  WORK SPACE                                 C
C        PREC               DIMENSION OF MP NUMBERS                    C
C        MAX                DIMENSION OF WORK SPACE                    C
C        PX                 DOUBLE PRECISION VALUE OF P                C
C        QX                 DOUBLE PRECISION VALUE OF Q                C
C        PQX                DOUBLE PRECISION VALUE OF PQ               C
C                                                                      C
C    THE SUBROUTINE MULLER COMPUTES THE VALUE OF I(P,Q;X) AS           C
C    FOLLOWS. LET G( ) BE THE GAMMA FUNCTION.                          C
C                                                                      C
C    I(P,Q;X)=(G(P+Q)/(G(P+1)*G(Q))) * (X**P) * ((1-X)**(Q-1)) *       C
C             (C(1)/(1+C(2)/(1+C(3)/(1+C(4)/(1+...........)            C
C                                                                      C
C    THE VALUE OF I(P,Q;X) IS COMPUTED IN TWO PARTS. FIRST             C
C                                                                      C
C       K=(G(P+Q)/(G(P+1)*G(Q)) * (X**P) * ((1-X)**(Q-1)))             C
C                                                                      C
C    IS FOUND, THEN THE CONTINUED FRACTION IS COMPUTED. THE VALUE      C
C    OF LN(K) IS COMPUTED USING THE LGAM AND LN SUBROUTINES, THEN      C
C    THE MEXP SUBROUTINE IS USED TO FIND K. THE C( )'S IN THE          C
C    CONTINUED FRACTION ARE DEFINED AS FOLLOWS:                        C
C       C(1)=1                                                         C
C       C(2N)=-((P+N-1)*(Q-N)/((P+2N-2)*(P+2N-1))) * (X/(1-X))         C
C       C(2N+1)=((N*(P+Q-1+N)/((P+2N-1)*(P+2N)) * (X/(1-X))            C
C    THE NTH CONVERGENT IS V(N)=P(N)/Q(N) WHERE                        C
C        P(N)=P(N-1)+C(N)*P(N-2)                                       C
C        Q(N)=Q(N-1)+C(N)*Q(N-2)                                       C
C        P(-1)=1                                                       C
C        P(0)=0                                                        C
C        Q(-1)=0                                                       C
```

```
C         Q(0)=1                                                        C
C     THE CONVERGENTS 2,3,6,7,10,11,... BOUND THE CONTINUED FRACTION    C
C     FROM ONE SIDE, WHILE THE CONVERGENTS 4,5,8,9,12,13,.... BOUND     C
C     THE CONTINUED FRACTION FROM THE OTHER SIDE UNTIL C(2N)>0.         C
C     WHEN C(2N)>0 SUCCESIVE CONVERGENTS BOUND THE CONTINUED FRACTION.  C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
        IMPLICIT INTEGER (A-Z)
        REAL*8 PX,QX,PQX
        DIMENSION P(PREC),Q(PREC),PQ(PREC),X(PREC),X1(PREC)
      1,W(MAX),IBETA(PREC)
       PREC2=2*PREC
       ND=PREC-2
       CBETA=1
       GAM=CBETA+PREC
       LNX=GAM
       WORK=GAM+PREC
        SDIM=36*PREC+300
        RSP=6*PREC+24
        DSP=8*PREC+80
        LDIM=21*PREC+288
        EDIM=20*PREC+200
C
C     FIND THE VALUE OF LN(K) AND STORE IN CBETA.
C
       CALL ASSIGN(IBETA,P,PREC)
       CALL ADD1(IBETA,PREC)
C
C     GAM CONTAINS -LN(G(P+1))
C
       CALL LGAM(IBETA,PX+1.D0,W(GAM),W(WORK),PREC,SDIM)
       W(GAM)=-1*W(GAM)
       CALL ASSIGN(W(CBETA),W(GAM),PREC)
C
C     FIND -LN(G(Q))
C
       CALL LGAM(Q,QX,W(GAM),W(WORK),PREC,SDIM)
       W(GAM)=-1*W(GAM)
       CALL ADD(W(CBETA),W(GAM),W(CBETA),PREC)
       CALL LGAM(PQ,PQX,W(GAM),W(WORK),PREC,SDIM)
C
C     FIND LN(G(P+Q))
C
       CALL ADD(W(CBETA),W(GAM),W(CBETA),PREC)
C
C     FIND P*LN(X)
C
       CALL LN(X,W(LNX),PREC,W(WORK),LDIM)
      CALL MULT(W(LNX),P,W(LNX),W(WORK),PREC,PREC2)
      CALL ADD(W(CBETA),W(LNX),W(CBETA),PREC)
C
C     FIND (Q-1)*LN(1-X)
C
      CALL LN(X1,W(LNX),PREC,W(WORK),LDIM)
      CALL ASSIGN(IBETA,Q,PREC)
      IBETA(1)=-1
      CALL ADD1(IBETA,PREC)
      IBETA(1)=-1*IBETA(1)
      CALL MULT(IBETA,W(LNX),W(LNX),W(WORK),PREC,PREC2)
      CALL ADD(W(CBETA),W(LNX),W(CBETA),PREC)
C
C     FIND K FROM LN(K)
C
      CALL MEXP(W(CBETA),W(CBETA),PREC,W(WORK),EDIM)
C
C     POINTER        DESCRIPTION
C       P1           P(N-2)
C       P2           P(N-1)
C       P3           P(N)
C       Q1           Q(N-2)
C       Q2           Q(N-1)
C       Q3           Q(N)
C       V1           V(N-2)
C       V2           V(N-1)
C       V3           V(N)
C       PRODP        (P+2N-2)*(P+2N-1)*(1-X) OR (P+2N-1)*(P+2N)*(1-X)
C                    DENOMINATOR OF C(2N) OR C(2N+1)
C       TWO          MP CONSTANT 2
C       PRODPQ       (P+N-1)*(Q-N) NUMERATOR OF C(2N)
C       TWOX         MP CONSTANT 2*X
C       QMPM4        (Q-P-2N)*X USED TO FIND THE NUMERATOR OF
C                    C(2N) FROM C(2N-2)
C       ODDN         N*(P+Q-1+N)*X NUMERATOR OF C(2N+1)
C       TWOP4        2P+4N OR 2P+4N-2 USED TO FIND THE DENOMINATOR
C                    OF C(2N+1) FROM C(2N) OR C(2N) FROM C(2N-1)
C       TWOP4X       (2P+4N)*X OR (2P+4N-2)*X USED LIKE TWOP4
C       PQ2R         (P+Q+2N)*X USED TO FIND THE NUMERATOR OF
C                    C(2N+1) FROM C(2N-1)
C       B            C(2N) OR C(2N+1)
C       WORK         WORK SPACE
C
      P1=GAM
      P2=WORK
      P3=P2+PREC
      Q1=P3+PREC
      Q2=Q1+PREC
      Q3=Q2+PREC
      V1=Q3+PREC
      V2=V1+PREC
      V3=V2+PREC
      PRODP=V3+PREC
      TWO=PRODP+PREC
      PRODPQ=TWO+PREC
      TWOX=PRODPQ+PREC
      QMPM4=TWOX+PREC
      ODDN=QMPM4+PREC
      TWOP4=ODDN+PREC
      B=TWOP4+PREC
      PQ2R=B+PREC
      TWOP4X=PQ2R+PREC
      WORK=TWOP4X+PREC
C
C     P(1)=1
C     P(2)=1
C     Q(1)=1
C
      CALL CONVI(W(P1),1,PREC)
      CALL ASSIGN(W(P2),W(P1),PREC)
      CALL ASSIGN(W(Q1),W(P1),PREC)
      CALL CONVI(W(TWO),2,PREC)
      CALL MULTI(X,-2,W(WORK),PREC,PREC+1)
      CALL ASSIGN(W(TWOX),W(WORK),PREC)
      CALL ASSIGN(W(PRODP),P,PREC)
      CALL ADD1(W(PRODP),PREC)
       CALL ASSIGN(W(TWOP4X),W(PRODP),PREC)
       CALL MULT(W(PRODP),X1,W(PRODP),W(WORK),PREC,PREC2)
      CALL ASSIGN(W(PRODPQ),Q,PREC)
      W(PRODPQ)=-1
      CALL ADD1(W(PRODPQ),PREC)
      CALL MULT(W(PRODPQ),X,W(PRODPQ),W(WORK),PREC,PREC2)
      CALL DIV(W(PRODPQ),W(PRODP),W(B),W(WORK),PREC,DSP)
C
C     FIND Q(2) AND V(2)
C
      CALL ADD(W(Q1),W(B),W(Q2),PREC)
      CALL RECIPT(W(Q2),W(V2),W(WORK),PREC,RSP)
      CALL ADD(W(PRODPQ),X,W(PRODPQ),PREC)
      W(PRODPQ)=-1*W(PRODPQ)
      CALL ASSIGN(W(QMPM4),W(PRODPQ),PREC)
      CALL MULT(W(PRODPQ),W(TWOP4X),W(PRODPQ),W(WORK),PREC,PREC2)
      CALL ADD(P,W(TWO),W(V3),PREC)
      CALL MULT(W(V3),W(PRODP),W(PRODP),W(WORK),PREC,PREC2)
      CALL MULT(X,PQ,W(ODDN),W(WORK),PREC,PREC2)
      W(TWOX)=1
      CALL ADD(W(ODDN),W(TWOX),W(PQ2R),PREC)
      CALL ADD(W(V3),W(V3),W(TWOP4),PREC)
      CALL MULT(X,W(TWOP4),W(TWOP4X),W(WORK),PREC,PREC2)
      W(TWOP4X)=-1
      W(TWOX)=-1
      P(1)=-1
      CALL MULT(P,X,W(V3),W(WORK),PREC,PREC2)
      CALL ADD(W(QMPM4),W(V3),W(QMPM4),PREC)
      P(1)=1
C
C     FIND C(3),P(3),Q(3), AND V(3)
C
      CALL DIV(W(ODDN),W(PRODP),W(B),W(WORK),PREC,DSP)
      CALL ADD(W(B),W(P2),W(P3),PREC)
      CALL ADD(W(B),W(Q2),W(Q3),PREC)
      CALL DIV(W(P3),W(Q3),W(V3),W(WORK),PREC,DSP)
      ALT=0
C
C   RESET POINTERS FOR NEXT CONVERGENT
C
90    C=P1
      P1=P2
      P2=P3
      P3=C
      C=Q1
      Q1=Q2
      Q2=Q3
      Q3=C
      C=V1
      V1=V2
      V2=V3
      V3=C
C
C   EVALUATE THE EVEN CONVERGENTS V(2N).
C
C    FIND THE DENOMINATOR OF C(2N). LET D( ) BE THE DENOMINATOR,
C    THEN D(C(2N))=D(C(2N-1))+ 2P+4N-4 - (2P+4N-4)*X OR
C    D(C(2N))=PRODP+TWOP4+TWOP4X. STORE D(C(2N)) IN PRODP.
C
      CALL ADD(W(PRODP),W(TWOP4),W(PRODP),PREC)
      CALL ADD(W(PRODP),W(TWOP4X),W(PRODP),PREC)
C
C     FIND C(2N)=PRODPQ/PRODP
C
      CALL DIV(W(PRODPQ),W(PRODP),W(B),W(WORK),PREC,DSP)
      W(B)=-1*W(B)
C
C     FIND P(2N),Q(2N), AND V(2N)
C
      CALL MULT(W(B),W(P1),W(P3),W(WORK),PREC,PREC2)
      CALL ADD(W(P3),W(P2),W(P3),PREC)
      CALL MULT(W(B),W(Q1),W(Q3),W(WORK),PREC,PREC2)
      CALL ADD(W(Q3),W(Q2),W(Q3),PREC)
      CALL DIV(W(P3),W(Q3),W(V3),W(WORK),PREC,DSP)
```

```
C
C      UPDATE TWOP4 AND TWOP4X SO THAT THE DENOMINATOR OF C(2N+1)
C      CAN BE FOUND.
C
       CALL ADD(W(TWOP4X),W(TWOX),W(TWOP4X),PREC)
       CALL ADD(W(TWOP4),W(TWO),W(TWOP4),PREC)
C
C     FIND THE NUMERATOR OF C(2N+2). LET N( ) BE THE NUMERATOR,
C     THEN N(C(2N+2))=N(C(2N)) + (Q-P-2N)*X OR N(C(2N+2)= PRODPQ +
C     QMPM4. UPDATE QMPM4. THEN FIND N(C(2N+2)) AND STORE IN PRODPQ.
C
       CALL ADD(W(TWOX),W(QMPM4),W(QMPM4),PREC)
       CALL ADD(W(PRODPQ),W(QMPM4),W(PRODPQ),PREC)
C
C     FIND IF SUFFICIENT ACCURACY HAS BEEN ACHIEVED.
C
       DO 100 J=1,ND
       IF (W(V3-1+J).NE.W(V2-1+J)) GOTO 110
100    CONTINUE
       GOTO 200
C
C     RESET THE POINTERS FOR THE NEXT CONVERGENT
C
110    C=P1
       P1=P2
       P2=P3
       P3=C
       C=Q1
       Q1=Q2
       Q2=Q3
       Q3=C
       C=V1
       V1=V2
       V2=V3
       V3=C
C
C     FIND THE ODD CONVERGENT V(2N+1)
C
C     LET N( ) BE THE NUMERATOR. THEN FIND N(C(2N+1) BY
C     N(C(2N+1))=N(C(2N-1))+(P+Q+2N)*X. HENCE N(C(2N+1))=
C     ODDN + PQ2R. STORE THE NEW NUMERATOR IN ODDN AND UPDATE
C     PQ2R=PQ2R+2*X.
C
       CALL ADD(W(ODDN),W(PQ2R),W(ODDN),PREC)
       W(TWOX)=1
       CALL ADD(W(PQ2R),W(TWOX),W(PQ2R),PREC)
       W(TWOX)=-1
```

```
C
C     FIND THE DENOMINATOR OF C(2N+1). LET D( ) BE THE
C     DENOMINATOR. THEN D(C(2N+1))=D(C(2N))+2P+4N-2-(2P+4N-2)*X
C     OR D(C(2N+1)=PRODP + TWOP4 + TWOP4X. STORE D(C(2N+1) IN
C     PRODP. UPDATE TWOP4 AND TWOP4X.
C
      CALL ADD(W(PRODP),W(TWOP4),W(PRODP),PREC)
      CALL ADD(W(TWOP4X),W(PRODP),W(PRODP),PREC)
      CALL ADD(W(TWOP4),W(TWO),W(TWOP4),PREC)
      CALL ADD(W(TWOP4X),W(TWOX),W(TWOP4X),PREC)
C
C    FIND C(2N+1),P(2N+1),Q(2N+1) AND V(2N+1)
C
      CALL DIV(W(ODDN),W(PRODP),W(B),W(WORK),PREC,DSP)
      CALL MULT(W(B),W(P1),W(P3),W(WORK),PREC,PREC2)
      CALL ADD(W(P2),W(P3),W(P3),PREC)
      CALL MULT(W(Q1),W(B),W(Q3),W(WORK),PREC,PREC2)
      CALL ADD(W(Q2),W(Q3),W(Q3),PREC)
      CALL DIV(W(P3),W(Q3),W(V3),W(WORK),PREC,DSP)
      IF(ALT.EQ.1) GOTO 130
C
C   FIND IF N(C(2N+2)>0 SO THAT THE APPROPRIATE TEST FOR
C   ACCURACY CAN BE USED.
C
      IF(W(PRODPQ).EQ.-1) ALT=1
C
C   TEST FOR DESIRED ACCURACY.
C
      DO 120 J=1,ND
      IF (W(V3-1+J).NE.W(V1-1+J)) GOTO 90
120   CONTINUE
      GOTO 200
C
C   TEST FOR DESIRED ACCURACY.
C
130   DO 140 J=1,ND
      IF(W(V3-1+J).NE.W(V2-1+J)) GOTO 90
140   CONTINUE
C
C   ASSEMBLE THE VALUE I(P,Q;X) AND RETURN
C
200   CALL MULT(W(V3),W(CBETA),IBETA,W(WORK),PREC,PREC2)
      RETURN
      END
      SUBROUTINE MULT(X,Y,Z,W,P,Q)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C     THE SUBROUTINE MULT(X,Y,Z,W,P,Q) MULTIPLIES THE MP NUMBERS       C
C     X AND Y. THE PRODUCT IS RETURNED IN THE MP NUMBER Z. THE         C
C     ALGORITHM USED TO MULTIPLE THE MANTISSAS OF X AND Y IS           C
C     GIVEN IN KNUTH(1963) THE ART OF COMPUTER PROGRAMMING VOL. 2      C
C     PAGE 233. THE MANTISSA OF X IS THE INTEGER U. THE MANTISSA OF    C
C     Y IS THE INTEGER V. THE ELEMENTS W(5),W(6),...,W(2*P)            C
C     CORRESPOND TO THE ARRAY W IN THE ALGORITHM.                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      INTEGER X,Y,Z,W,P,Q
      DIMENSION X(P),Y(P),Z(P),W(Q)
      DO 10 J=P,Q
        W(J)=0
10    CONTINUE
      J=P
20    IF (Y(J).EQ.0) GOTO 40
        I=P
        K=0
30      ITOT=X(I)*Y(J)+W(I+J)+K
        K=ITOT/10000
        W(I+J)=ITOT-K*10000
        I=I-1
        IF (I.GE.3) GOTO 30
          W(J+2)=K
          GOTO 50
40    W(J+2)=0
50    J=J-1
      IF (J.GE.3) GOTO 20
C
C    FIND THE SIGN AND EXPONENT OF THE PRODUCT
C
        Z(1)=X(1)*Y(1)
        Z(2)=X(2)+Y(2)
        IPOS=2
C
C    FIND THE MANTISSA OF Z AND ADJUST THE EXPONENT IF NECESSARY
C
        IF (W(5).EQ.0) IPOS=3
        DO 60 I=3,P
          Z(I)=W(I+IPOS)
60      CONTINUE
        IF(IPOS.EQ.3) Z(2)=Z(2)-1
        RETURN
      END
      SUBROUTINE MULTI(X,I1,W,D1,D2)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C    THE SUBROUTINE MULTI(X,I1,W,D1,D2) MULTIPLIES THE MP NUMBER X BY  C
C    THE INTEGER NUMBER I1. THE RESULTING PRODUCT IS RETURNED IN THE   C
C    WORK ARRAY W.                                                     C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      INTEGER X,W,D1,D2
      DIMENSION X(D1),W(D2)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C           IF THE MP NUMBER X IS ZERO RETURN.                         C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IF (X(1).NE.0) GOTO 10
        W(1)=0
        RETURN
10    DO 15 J=D1,D2
        W(J)=0
15    CONTINUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C    INCREASE THE EXPONENT OF THE MP NUMBER X BY 1, SINCE THE          C
C    RESULTING PRODUCT IS IN UNNORMALIZED FORM. THEN FIND THE          C
C    SIGN OF THE PRODUCT.                                              C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      W(2)=X(2)+1
      W(1)=X(1)*ISIGN(1,I1)
      I2=IABS(I1)
      J=D1
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C    LOOP AND FIND THE PRODUCT OF EACH EXTENDED DIGIT WITH THE         C
C    INTEGER. AFTER EACH MULTIPLICATION DETERMINE THE CARRY.           C
C    CONTINUE LOOPING UNTIL ALL DIGITS OF THE MP NUMBER HAVE           C
C    MULTIPLIED.                                                       C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

```
20     IF (X(J).NE.0) GOTO  30
         W(J)=0
         GO TO 50
30     IT=I2*X(J)+W(J+1)
       K=IT/10000
       W(J+1)=IT-10000*K
       W(J)=K
50     J=J-1
       IF (J.GE.3) GOTO 20
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C    IF THE LAST MULTIPLICATION DID NOT RESULT IN A CARRY BEING       C
C    GENERATED, NORMALIZE THE MP NUMBER W AND RETURN.                 C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
         IF (W(3).GE.10000) GOTO 55
           CALL VNORM(W,D2)
           RETURN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C     A CARRY HAS BEEN GENERATED IN THE LAST MULTIPLICATION. SHIFT    C
C     THE DIGITS OF W DOWN AND DETERMINE THE CARRY. PLACE THE CARRY   C
C     IN THE HIGHEST ORDER DIGIT. UPDATE THE EXPONENT AND RETURN.     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
55       P3=D1+3
         DO 60 J=3,D1
           POS=P3-J
           W(POS+1)=W(POS)
60       CONTINUE
         W(3)=W(4)/10000
         W(4)=W(4)-W(3)*10000
         W(2)=W(2)+1
         RETURN
         END
```

```
      SUBROUTINE RECIPT(Z,X,W,P,MAX)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C                                                                       C
C     THE SUBROUTINE RECIPT(Z,X,W,P,MAX) FINDS THE RECIPROCAL OF THE    C
C     MP NUMBER Z AND RETURNS THE RECIPROCAL IN THE MP NUMBER X. FOUR   C
C     GAURD DIGITS ARE USED TO PREVENT ROUND OFF ERROR. THE METHOD      C
C     EMPLOYED TO FIND THE RECIPROCAL IS AN ITERATIVE THIRD ORDER       C
C     NEWTON'S METHOD. THE APPROXIMATION AT THE ITH ITERATION IS        C
C     FOUND AS  X(I)=X(I-1)*(1+(1-Z*X(I-1))*(1+(1-Z*X(I-1)))).          C
C     TO OPTIMIZE EXECUTION TIME THE NUMBER OF DIGITS CARRIED ON        C
C     EACH ITERATION IS TRIPLED, THE FIRST APPROXIMATION USES 2         C
C     DIGITS. THE ITERATION IS CARRIED ON UNTIL THE DESIRED NUMBER      C
C     OF CORRECT DIGITS ARE ACHIEVED. THE EXPONENT OF Z IS SET TO       C
C     ONE AND THE RECIPROCAL FOUND. THE EXPONENT IS THEN CORRECTED      C
C     AND THE RECIPROCAL RETURNED IN X.                                 C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION Z(P),X(P),W(MAX)
      K=P+4
      K1=K+1
      CALL CONVI(W(1),1,K)
      DO 10 I=3,P
        W(K+I)=Z(I)
10    CONTINUE
C
C     W(K1) CONTAINS THE VALUE OF Z WITH THE EXPONENT SET TO 1
C
      W(K1)=1
      W(K1+1)=1
      K2=K1+K
      W(K2-1)=0
      W(K2-2)=0
      W(K2-3)=0
      W(K2-4)=0
      K3=K2+K
      K4=K3+K
C
C     W(K2) CONTAINS X(0) THE FIRST APPROXIMATION OF (1/Z)
C
      CALL DIVI(W(1),Z(3),W(K2),6,6)
C
C     W(1) WILL CONTAIN THE APPROXIMATIONS X(I)
C
      CALL ASSIGN(W(1),W(K2),6)
      D1=2
```

```
C
C     FIND THE NUMBER OF DIGITS TO USE ON THIS ITERATION
C
15     D=MIN0(3*D1,K-2)
       IF (D.EQ.K-2) GOTO 17
         D2=D1+3
         D1=D
         D3=D+2
         DO 16 I=D2,D3
           W(I)=0
16       CONTINUE
17     ND=D+2
C
C     W(K2) IS SET TO  Z*X(I-1)
C
       CALL MULT(W(1),W(K1),W(K2),W(K4),ND,2*ND)
       W(K2)=W(K2)*(-1)
C
C     W(K2) IS   1-Z*X(I-1)
C
       CALL ADD1(W(K2),ND)
       CALL ASSIGN(W(K3),W(K2),ND)
C
C     W(K3) IS   1+(1-Z*X(I-1))
C
       CALL ADD1(W(K3),ND)
C
C     W(K2) IS   (1-Z*X(I-1))*(1+(1-Z*X(I-1)))
C
       CALL MULT(W(K2),W(K3),W(K2),W(K4),ND,2*ND)
C
C     W(K2) IS   (1+(1-Z*X(I-1))*(1+(1-Z*X(I-1))))
C
       CALL ADD1(W(K2),ND)
C
C     W(K2) IS X(I)=X(I-1)*(1+(1-Z*X(I-1))*(1+(1-Z*X(I-1))))
C
       CALL MULT(W(1),W(K2),W(K2),W(K3),ND,2*ND)
       IF (D.NE.K-2) GOTO 40
       DO 20 I=1,P
C
C     CHECK TO SEE IF DESIRED NUMBER OF DIGITS ACHIEVED
C
         IF (W(I)-W(K2-1+I)) 40,20,40
20     CONTINUE
```

```
C
C    ASSIGN X THE VALUE OF 1/Z WHEN THE EXPONENT OF Z HAS BEEN
C    SET TO 1. THEN ADJUST THE EXPONENT TO FIND THE VALUE OF
C    OF 1/Z.
C
      DO 30 I=3,P
        X(I)=W(I)
30    CONTINUE
      X(1)=Z(1)
      X(2)=W(2)-Z(2)+1
      RETURN
C
C    ITERATE AGAIN AS THE DESIRED ACCURAY HAS NOT BEEN ACHIEVED
C
40    CALL ASSIGN(W(1),W(K2),ND)
      GO TO 15
      END
      SUBROUTINE RPLQ(P,Q,X,PQ,X1,IBETA,W,PREC,MAX,PX,QX,PQX)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C    THE SUBROUTINE RPLQ COMPUTES THE VALUE OF I(P,Q;X). TWO           C
C    CONTINUED FRACTIONS ARE USED TO FIND THE VALUE OF I(P,Q;X).       C
C                                                                      C
C    SUBROUTINE PARAMETER   DESCRIPTION                                C
C        P                  MP NUMBER CONTAINING THE VALUE OF P        C
C                           IN I(P,Q;X)                                C
C        Q                  MP NUMBER CONTAINING THE VALUE OF Q        C
C                           IN I(P,Q;X)                                C
C        X                  MP NUMBER CONTAINING THE VALUE OF X        C
C                           IN I(P,Q;X)                                C
C        PQ                 MP NUMBER WITH THE VALUE P+Q               C
C        X1                 MP NUMBER CONTAINING 1-X                   C
C        IBETA              MP NUMBER RETURNED WITH THE VALUE          C
C                           I(P,Q;X)                                   C
C        W                  WORK SPACE                                 C
C        PREC               DIMENSION OF MP NUMBERS                    C
C        MAX                DIMENSION OF WORK SPACE                    C
C        PX                 DOUBLE PRECISION VALUE OF P                C
C        QX                 DOUBLE PRECISION VALUE OF Q                C
C        PQX                DOUBLE PRECISION VALUE OF PQ               C
C                                                                      C
C    LET G( ) BE THE GAMMA FUNCTION AND INT( ) BE THE INTEGER          C
C    PART OF A NUMBER.THEN                                             C
C                                                                      C
C    I(P,Q;X)=(G(P+Q)/(G(P)*G(Q+1))) * (X**P) * ((1-X)**(Q-1))         C
C              * (F/(1-F)) + I(P+INT(Q),Q-INT(Q);X) .                  C
C                                                                      C
C    WHERE F=C(1)/(1+C(1)-C(2)/(1+C(2)-C(3)/(1+C(3)-............        C
C           -C(INT(Q)/(1+C(INT(Q)))........)                           C
C                                                                      C
C          C(1)=Q/P                                                    C
C          C(N)= ((Q-N+1)*X)/((P+N-1)*(1-X)) FOR N=2,...,INT(Q)        C
C                                                                      C
C    IF F(N) IS THE NTH CONVERGENT THEN F(N)/(1-F(N))=P(N)             C
C    WHERE P(N)=P(N-1)+C(N)*(P(N-1)-P(N-2)), P(-1)=1, AND P(0)=0.      C
C    LET E(N) BE THE TRUNCATION ERROR IF P(N) IS USED TO FIND          C
C    AN APPROXIMATION TO I(P,Q;X). DEFINE                              C
C        B(1)=(G(P+Q)/(G(P+2)*G(Q-1))) * (X**(P+1)) * ((1-X)**(Q-2))   C
C        B(N)=B(N-1)*C(N+1)  FOR N=2,......,INT(Q)-2                   C
C    THEN                                                              C
C        B(N)<E(N)<B(N)*(1/(1-C(N+2)))                                 C
C    IF E(N) IS  BOUNDED SUFFICIENTLY SMALL THE SUBROUTINE RPLQ        C
C    RETURNS THE APPROXIMATION BASED ON P(N). IF E(INT(Q)-2) IS        C
C    NOT SUFFICIENTLY SMALL P(INT(Q)) IS FOUND AND A CONTINUED         C
C    FRACTION IS USED TO EVALUATE I(P+INT(Q),Q-INT(Q);X).              C
```

```
C     LET P'=P+INT(Q) AND Q'=Q-INT(Q).                                  C
C                                                                       C
C     I(P',Q';X)=(G(P'+Q'-1)/(G(P'-1)*G(Q')) * (X**P') * ((1-X)**Q')    C
C                * (F/(1-F))                                            C
C     WHERE F=D(1)/(1+D(1)-D(2)/(1+D(2)-D(3)/(1+D(3)-............       C
C           D(1)=(P'+Q')/P'                                             C
C           D(N)=((P'+Q'+N-1)*X)/(P'+N-1) FOR N=2,3,.........           C
C                                                                       C
C     IF F(N) IS THE NTH CONVERGENT THEN F(N)/(1-F(N))=P(N)             C
C     WHERE P(N)=P(N-1)+D(N)*(P(N-1)-P(N-2)), P(-1)=1, AND P(0)=0.      C
C                                                                       C
C     LET E(N) BE THE TRUNCATION ERROR IF P(N) IS USED TO FIND          C
C     AN APPROXIMATION TO I(P',Q';X). DEFINE                            C
C         B(1)=(G(P'+Q')/(G(P'+1)*G(Q'))) * (X**P') * ((1-X)**Q')       C
C         B(N)=B(N-1)*D(N+1)  FOR N=2,......                            C
C     THEN                                                              C
C         B(N)*(1/(1-X))<E(N)<B(N)*(1/(1-D(N+2)))                       C
C                                                                       C
C     WHEN E(N) IS SUFFICIENTLY SMALL AN APPROXIMATION USING           C
C     P(N) IS FOUND TO I(P',Q';X) AND THE APPROXIMATION TO              C
C     I(P,Q;X) IS ASSEMBLED AND RETURNED.                               C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
        IMPLICIT INTEGER (A-Z)
        REAL*8 PX,QX,PQX
        DIMENSION P(PREC),Q(PREC),PQ(PREC),X(PREC),X1(PREC)
     1,W(MAX),IBETA(PREC)
       PREC2=2*PREC
       ND=PREC-2
       CBETA=1
       GAM=CBETA+PREC
       LNX=GAM
       WORK=GAM+PREC
        SDIM=36*PREC+300
        RSP=6*PREC+24
        DSP=8*PREC+80
        LDIM=21*PREC+288
        EDIM=20*PREC+200
C
C     FIND LN(K), K=(G(P+Q)/(G(P)*G(Q+1))*(X**P)*((1-X)**(Q-1))
C
C     FIND LN(G(P))
C
       CALL LGAM(P,PX,W(GAM),W(WORK),PREC,SDIM)
       W(GAM)=-1*W(GAM)
       CALL ASSIGN(W(CBETA),W(GAM),PREC)
```

```
C
C     FIND LN(G(Q+1))
C
        CALL ASSIGN(IBETA,Q,PREC)
        CALL ADD1(IBETA,PREC)
       CALL LGAM(IBETA,QX+1.D0,W(GAM),W(WORK),PREC,SDIM)
       W(GAM)=-1*W(GAM)
       CALL ADD(W(CBETA),W(GAM),W(CBETA),PREC)
C
C     FIND LN(G(P+Q))
C
       CALL LGAM(PQ,PQX,W(GAM),W(WORK),PREC,SDIM)
       CALL ADD(W(CBETA),W(GAM),W(CBETA),PREC)
C
C     FIND P*LN(X)
C
       CALL LN(X,W(LNX),PREC,W(WORK),LDIM)
       CALL MULT(W(LNX),P,W(LNX),W(WORK),PREC,PREC2)
       CALL ADD(W(CBETA),W(LNX),W(CBETA),PREC)
C
C     FIND (Q-1)*LN(1-X)
C
       CALL LN(X1,W(LNX),PREC,W(WORK),LDIM)
       CALL ASSIGN(IBETA,Q,PREC)
       IBETA(1)=-1
       CALL ADD1(IBETA,PREC)
       IBETA(1)=-1*IBETA(1)
       CALL MULT(IBETA,W(LNX),W(LNX),W(WORK),PREC,PREC2)
       CALL ADD(W(CBETA),W(LNX),W(CBETA),PREC)
C
C     FIND K FROM LN(K) AND STORE IN CBETA
C
       CALL MEXP(W(CBETA),W(CBETA),PREC,W(WORK),EDIM)
C
C     POINTER      DESCRIPTION
C       P1           P(N-2)
C       P2           P(N-1)
C       DELTAP       P(N-1)-P(N-2)  OR P(N)
C       CPRM         B(N)
C       B            C(N)
C       R            C(N+2) OR 1/(1-C(N+2))
C       PPN          P+N
C       QMNX         (Q-N+1)*X
C       PPN1MX       (P+N-1)*(1-X)
C       QMN          (Q-N)
C       XPQ          (P+Q+N-1)*X
C       BNEXT        C(N+1)
C       WORK         WORK SPACE
C
       P1=GAM
```

```
      P2=WORK
       DELTAP=P2+PREC
       CPRM=DELTAP+PREC
       B=CPRM+PREC
       R=B+PREC
       PPN=R+PREC
       QMNX=PPN+PREC
       PPN1MX=QMNX+PREC
       QMN=PPN1MX+PREC
       XPQ=QMN+PREC
       BNEXT=XPQ+PREC
       WORK=BNEXT+PREC
C
C     EVALUATE I(P,Q;X) USING THE FIRST CONTINUED FRACTION
C     WITH C(1)=Q/P, C(N)=((Q-N+1)*X)/((P+N-1)*(1-X)) FOR
C     N=2,...,INT(Q).
C
C       FIND B(1) FOR THE FIRST CONTINUED FRACTION
C
       CALL ASSIGN(W(CPRM),W(CBETA),PREC)
       CALL MULT(W(CPRM),Q,W(CPRM),W(WORK),PREC,PREC2)
       CALL DIV(W(CPRM),P,W(CPRM),W(WORK),PREC,DSP)
       CALL ASSIGN(W(QMN),Q,PREC)
      IBETA(1)=0
       CALL ASSIGN(W(PPN),P,PREC)
       CALL ADD1(W(PPN),PREC)
       CALL MULT(PQ,X,W(XPQ),W(WORK),PREC,PREC2)
C
C     CHECK IF INT(Q)>=1
C
       W(QMN)=-1
       CALL ADD1(W(QMN),PREC)
       IF (W(QMN).EQ.1) GOTO 30
C
C     FIND P(1)
C
       CALL DIV(Q,P,W(P1),W(WORK),PREC,DSP)
       CALL MULT(W(CBETA),W(P1),IBETA,W(WORK),PREC,PREC2)
C
C     Q=1 AND THE CONTINUED FRACTION HAS BEEN COMPUTED. RETURN
C     THE VALUE I(P,Q;X).
C
        IF (W(QMN).EQ.0) RETURN
       CALL ASSIGN(W(QMNX),W(QMN),PREC)
       W(QMNX)=-1*W(QMNX)
       CALL MULT(W(QMNX),X,W(QMNX),W(WORK),PREC,PREC2)
       CALL MULT(P,X1,W(PPN1MX),W(WORK),PREC,PREC2)
       CALL ADD(W(PPN1MX),X1,W(PPN1MX),PREC)
```

```
C
C     FIND C(2)
C
        CALL DIV(W(QMNX),W(PPN1MX),W(B),W(WORK),PREC,DSP)
C
C     FIND B(2)
C
        CALL MULT(W(CPRM),W(B),W(CPRM),W(WORK),PREC,PREC2)
        CALL ADD1(W(PPN),PREC)
C
C     CHECK INT(Q)>=2
C
        CALL ADD1(W(QMN),PREC)
        IF (W(QMN).EQ.1) GOTO 30
C
C     FIND P(2)
C
        CALL ADD1(W(B),PREC)
        CALL MULT(W(P1),W(B),W(P2),W(WORK),PREC,PREC2)
C
C     FIND C(3)
C
        CALL ADD(W(PPN1MX),X1,W(PPN1MX),PREC)
        X(1)=-1
        CALL ADD(W(QMNX),X,W(QMNX),PREC)
        X(1)=1
        CALL DIV(W(QMNX),W(PPN1MX),W(B),W(WORK),PREC,DSP)
C
C     FIND B(N) FOR N=2,....
C
10      CALL MULT( W(CPRM),W(B),W(CPRM),W(WORK),PREC,PREC2)
C
C     FIND C(N+2) FOR N=2,....,INT(Q)-2
C
        CALL ADD(W(PPN1MX),X1,W(PPN1MX),PREC)
        X(1)=-1
        CALL ADD(W(QMNX),X,W(QMNX),PREC)
        X(1)=1
        CALL DIV(W(QMNX),W(PPN1MX),W(R),W(WORK),PREC,DSP)
        CALL ASSIGN(W(BNEXT),W(R),PREC)
C
C     FIND AN APPROXIMATION TO I(P,Q;X) USING P(N), AND ERROR E(N)
C
        W(R)=-1
        CALL ADD1(W(R),PREC)
        CALL RECIPT(W(R),W(R),W(WORK),PREC,RSP)
        W(R)=-1
        CALL ADD1(W(R),PREC)
        CALL MULT(W(CPRM),W(R),W(R),W(WORK),PREC,PREC2)
        CALL MULT(W(CBETA),W(P2),IBETA,W(WORK),PREC,PREC2)
```

```
C
C     FIND IF THE TRUNCATION ERROR IS SUFFICIENTLY SMALL
C
        IF ((IBETA(2)-W(R+1)).GT.ND) RETURN
C
C     IF INT(Q)=N THEN THE FRACTION HAS TERMINATED. RETURN
C     I(P,Q;X).
C
         IF (W(QMN).EQ.0) RETURN
C
C     CHECK THAT Q-N-1>=1
C
        CALL ADD1(W(QMN),PREC)
        IF (W(QMN).EQ.1) GOTO 30
C
C     FIND P(N+1)
C
        W(P1)=-1*W(P1)
        CALL ADD(W(P1),W(P2),W(DELTAP),PREC)
        CALL MULT(W(B),W(DELTAP),W(DELTAP),W(WORK),PREC,PREC2)
        CALL ADD(W(P2),W(DELTAP),W(DELTAP),PREC)
        W(P1)=-1*W(P1)
C
C     RESET POINTERS TO EVALUATE THE NEXT CONVERGENT
C
        C=P1
        P1=P2
        P2=DELTAP
        DELTAP=C
        CALL ASSIGN(W(B),W(BNEXT),PREC)
        GOTO 10
C
C     THE FIRST CONTINUED FRACTION HAS NOT SUPPLIED SUFFICIENT
C     ACCURACY. USE THE SECOND CONTINUED FRACTION TO EVALUATE
C     I(P+INT(Q),Q-INT(Q);X).
C
C     POINTER      DESCRIPTION
C       PQN          P+Q-1
C       RPSUM        VALUE OF FIRST CONTINUED FRACTION PLUS
C                    THE VALUE OF THE APPROXIMATION OF THE
C                    SECOND CONTINUED FRACTION
C
30      PQN=QMN
        RPSUM=QMNX
C
C     LET P'=P+INT(Q) AND Q'=Q-INT(Q).
C     FIND (G(P'+Q')/(G(P')*G(Q')))*(X**P')*((1-X)**Q')
C
        CALL MULT(W(CPRM),X1,W(CPRM),W(WORK),PREC,PREC2)
        CALL ASSIGN(W(CBETA),W(CPRM),PREC)
```

```
      CALL ASSIGN(W(RPSUM),IBETA,PREC)
      CALL ASSIGN(W(PQN),PQ,PREC)
      W(PPN)=-1
      CALL ADD1(W(PPN),PREC)
      W(PPN)=-1*W(PPN)
      W(PQN)=-1*W(PQN)
      CALL ADD1(W(PQN),PREC)
      W(PQN)=-1*W(PQN)
C
C    FIND D(1)
C
      CALL DIV(W(PQN),W(PPN),W(B),W(WORK),PREC,DSP)
C
C    FIND P(1)
C
      CALL ASSIGN(W(P1),W(B),PREC)
C
C    FIND APPROXIMATION OF I(P,Q;X) USING P(1)
C
      CALL ADD(W(CBETA),IBETA,IBETA,PREC)
C
C    FIND (G(P'+Q'-1)/(G(P'-1)*G(Q')))*(X**P')*((1-X)**Q')
C
      CALL DIV(W(CBETA),W(B),W(CBETA),W(WORK),PREC,DSP)
C
C    FIND D(2)
C
      CALL ADD1(W(PPN),PREC)
      CALL DIV(W(XPQ),W(PPN),W(B),W(WORK),PREC,DSP)
C
C    FIND D(3)
C
      CALL ADD(W(XPQ),X,W(XPQ),PREC)
      CALL ADD1(W(PPN),PREC)
      CALL DIV(W(XPQ),W(PPN),W(R),W(WORK),PREC,DSP)
C
C   FIND B(1)
C
      CALL MULT(W(CPRM),W(B),W(CPRM),W(WORK),PREC,PREC2)
      CALL ASSIGN(W(BNEXT),W(R),PREC)
C
C   CHECK TO SEE IF DESIRED ACCURACY ACHIEVED
C
      W(R)=-1
      CALL ADD1(W(R),PREC)
      CALL RECIPT(W(R),W(R),W(WORK),PREC,RSP)
      CALL MULT(W(R),W(CPRM),W(R),W(WORK),PREC,PREC2)
      IF ((IBETA(2)-W(R+1)).GT.ND) RETURN
```

```
C
C   FIND P(2)
C
      CALL ADD1(W(B),PREC)
      CALL MULT(W(P1),W(B),W(P2),W(WORK),PREC,PREC2)
C
C   FIND APPROXIMATION OF I(P,Q;X) USING P(2)
C
      CALL MULT(W(CBETA),W(P2),IBETA,W(WORK),PREC,PREC2)
      CALL ADD(W(RPSUM),IBETA,IBETA,PREC)
C
C   FIND B(2)
C
      CALL MULT(W(CPRM),W(BNEXT),W(CPRM),W(WORK),PREC,PREC2)
      CALL ASSIGN(W(B),W(BNEXT),PREC)
C
C   FIND D(4)
C
      CALL ADD(W(XPQ),X,W(XPQ),PREC)
      CALL ADD1(W(PPN),PREC)
      CALL DIV(W(XPQ),W(PPN),W(R),W(WORK),PREC,DSP)
C
C   CHECK TO SEE IF DESIRED ACCURACY ACHIEVED
C
      CALL ASSIGN(W(BNEXT),W(R),PREC)
      W(R)=-1
      CALL ADD1(W(R),PREC)
      CALL RECIPT(W(R),W(R),W(WORK),PREC,RSP)
      CALL MULT(W(CPRM),W(R),W(R),W(WORK),PREC,PREC2)
      IF ((IBETA(2)-W(R+1)).GT.ND) RETURN
C
C   FIND P(N) FOR N=3,4,........
C
100   W(P1)=-1*W(P1)
      CALL ADD(W(P2),W(P1),W(DELTAP),PREC)
      CALL MULT(W(B),W(DELTAP),W(DELTAP),W(WORK),PREC,PREC2)
      CALL  ADD(W(P2),W(DELTAP),W(DELTAP),PREC)
C
C   FIND APPROXIMATION OF I(P,Q;X) USING P(N)
C
      CALL MULT(W(CBETA),W(DELTAP),IBETA,W(WORK),PREC,PREC2)
      CALL ADD(W(RPSUM),IBETA,IBETA,PREC)
C
C   FIND B(N) N=3,4,....
C
      CALL MULT(W(CPRM),W(BNEXT),W(CPRM),W(WORK),PREC,PREC2)
      CALL ASSIGN(W(B),W(BNEXT),PREC)
```

```
C
C   FIND D(N+2) FOR N=3,4,.....
C
       CALL  ADD1(W(PPN),PREC)
       CALL ADD(W(XPQ),X,W(XPQ),PREC)
       CALL DIV(W(XPQ),W(PPN),W(R),W(WORK),PREC,DSP)
       CALL ASSIGN(W(BNEXT),W(R),PREC)
C
C   CHECK TO SEE IF DESIRED ACCURACY IS ACHIEVED.
C
       W(R)=-1
       CALL ADD1(W(R),PREC)
       CALL RECIPT(W(R),W(R),W(WORK),PREC,RSP)
       CALL MULT(W(CPRM),W(R),W(R),W(WORK),PREC,PREC2)
       IF ((IBETA(2)-W(R+1)).GT.ND) RETURN
C
C   RESET POINTERS TO EVALUATE THE NEXT CONVERGENT
C
       C=P1
       P1=P2
       P2=DELTAP
       DELTAP=C
       GOTO 100
       END
```

```
      SUBROUTINE UADD(X,Y,Z,P,NORM)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C    THE SUBROUTINE UADD(X,Y,Z,P,NORM) ADDS THE MP NUMBERS X AND Y     C
C    WHEN X AND  Y HAVE THE SAME SIGN BUT UNEQUAL EXPONENTS. THE       C
C    RESULT OF THE ADDITION IS RETURNED IN Z. THE VARIABLE NORM        C
C    CONTAINS THE DIFFERENCE OF THE EXPONENT OF Y FROM THE EXPONENT    C
C    OF X. THE NUMBER X IS LARGER THAN THE NUMBER Y IN ABSOLUTE        C
C    VALUE.                                                            C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION X(P),Y(P),Z(P)
C
C    IF THERE ARE NO OVERLAPPING DIGITS TO ADD, ASSIGN Z THE
C    VALUE OF X AND RETURN.
C
      IF (-NORM.LT.P-2) GOTO 10
        CALL ASSIGN(Z,X,P)
        RETURN
C
C    FIND THE NUMBER OF OVERLAPPING DIGITS.
C
10    OLAP=P-2+NORM
      ICARRY=0
      P1=P+1
C
C    LOOP ADDING THE OVERLAPPING DIGITS AND GENERATING CARRYS.
C
      DO 30 J=1,OLAP
        POS=P1-J
        ITOT=X(POS)+Y(POS+NORM)+ICARRY
        IF (ITOT.GE. 10000) GOTO 20
          Z(POS)=ITOT
          ICARRY=0
          GOTO 30
20      Z(POS)=ITOT-10000
        ICARRY=1
30    CONTINUE
C
C    IF A ZERO CARRY WAS GENERATED ON THE ADDITION OF THE LAST
C    OVERLAPPING DIGITS. ASSIGN THE SUM TO Z AND RETURN.
C
      IF (ICARRY.EQ.1) GOTO 40
        CALL ASSIGN(Z,X,POS-1)
        RETURN
```

```
C
C    CONTINUE THE ADDITION UNTIL A ZERO CARRY IS GENERATED
C
40    POS=POS-1
      ITOT=X(POS)+1
      IF (ITOT.EQ.10000) GOTO 50
        Z(POS)=ITOT
        CALL ASSIGN(Z,X,POS-1)
        RETURN
50    Z(POS)=0
      IF (POS.GT.3) GOTO 40
C
C    THE SUM IS OF LARGER MAGNITUDE THAN X, HENCE THE EXPONENT
C    MUST BE ADJUSTED AND THE DIGITS OF Z SHIFTED. THE NEW
C    LEADING DIGIT IS 1.
C
      P3=P-3
      DO 60  J=1,P3
        POS=P1-J
        Z(POS)=Z(POS-1)
60    CONTINUE
      Z(1)=X(1)
      Z(2)=X(2)+1
      Z(3)=1
      RETURN
      END
      SUBROUTINE UNADD(X,Y,Z,P,NORM)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C     THE SUBROUTINE UNADD(X,Y,Z,P,NORM) SUBTRACTS THE MP NUMBER Y     C
C     FROM THE MP NUMBER X WHEN X AND Y HAVE UNEQUAL EXPONENTS.        C
C     THE RESULT IS RETURNED IN THE MP NUMBER Z. THE NUMBER X          C
C     IS LARGER IN ABSOLUTE VALUE THAN THE NUMBER Y. THE VARIABLE      C
C     NORM CONTAINS THE DIFERENCE OF THE EXPONENT OF Y FROM THE        C
C     EXPONENT OF X. THIS SUBROUTINE IS CALLED BY THE SUBROUTINE ADD.  C
C     TWO GAURD DIGITS ARE USED IN THIS SUBTRACTION SUBROUTINE.        C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION X(P),Y(P),Z(P)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     FIND THE NUMBER OF EXTENDED DIGITS THAT OVERLAP WHEN Y IS        C
C     SHIFTED TO HAVE THE SAME EXPONENT AS X. IF THERE IS AN           C
C     OVERLAP BRANCH TO USE THE USUAL SUBTRACTION ALGORITHM.           C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      DP=P-2+NORM
      IF(DP)10,10,60
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C    THE TWO NUMBERS HAVE NO DIGITS IN COMMON. IF THE LAST DIGIT OF    C
C    X IS NON ZERO BRANCH TO SUBTRACT THE BORROW AND RETURN. OTHERWISE C
C    FIND THE BORROW GENERATED  BY THE TWO GAURD DIGITS AND PREFORM    C
C    THE SUBTRACTION DICTATED BY THE GAURD DIGITS.                     C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
10        IF (X(P).EQ.0) GOTO 15
          CALL ASSIGN(Z,X,P)
          Z(P)=Z(P)-1
      RETURN
15        DO 20 J=4,P
            IF (Y(J).GT.0) GOTO 25
20        CONTINUE
          IBRW=0
          GOTO 27
25        IBRW=1
27        G=Y(3)+IBRW
```

```
        P2=P+3
        DO 50 J=3,P
          POS=P2-J
          ITOT=X(POS)-1
C
C     IF A BORROW IS NOT GENERATED BRANCH TO RETURN THE VALUE Z
C
          IF (ITOT.GE.0) GOTO 40
            Z(POS)=9999
30      CONTINUE
40      CALL ASSIGN(Z,X,POS)
        Z(POS)=ITOT
        IF (POS.GT.3) RETURN
        IF (ITOT.GT.0) RETURN
C
C   A DIGIT HAS BEEN LOSS IN THE SUBTRACTION. THE GAURD DIGITS WILL
C   BE USED TO RECOVER THE LOST DIGIT.
C
          IP1=P-1
          DO 50 J1=3,IP1
            Z(J1)=Z(J1+1)
50        CONTINUE
C
C   IF THE TWO NUMBERS MISS OVERLAPPING BY ONE EXTENDED DIGIT,
C   THE GAURD DIGITS WILL BE USED TO FILL THE LAST DIGIT. OTHERWISE
C   THE LAST DIGIT WILL HAVE THE VALUE 9999.
C
          IF (DP.EQ.0) GOTO 55
            Z(P)=9999
            Z(2)=Z(2)-1
          RETURN
55        Z(P)=10000-G
          Z(2)=Z(2)-1
        RETURN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C    THE NUMBERS X AND Y HAVE OVERLAPPING DIGITS. GP IS THE SECOND     C
C    DIGIT OF Y NOT OVERLAPPING A DIGIT OF X. GP IS USED TO FIND       C
C    THE VALUE OF THE SECOND GAURD DIGIT.                              C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
60    GP=P+NORM+2
      IF (GP.GT.P) GOTO 75
        DO 70 J=GP,P
          IF (Y(J).NE.0) GOTO 80
70      CONTINUE
```

```
C
C    EITHER ONLY ONE DIGIT DOES NOT OVERLAP OR THE DIGITS OF Y
C    AFTER THE FIRST GAURD DIGIT ARE ZERO, HENCE ONLY ONE GAURD
C    DIGIT IS AVAILABLE.
C
75    IBRW=0
      GOTO 90
80    IBRW=1
C
C    FIND THE VALUE OF THE FIRST GAURD DIGIT.
C
90    G=10000-Y(GP-1)-IBRW
      IBRW=0
C
C    THE GAURD DIGITS ARE NON ZERO, HENCE A BORROW IS GENERATED.
C
      IF (G.NE.10000) IBRW=1
C
C    LOOP PREFORMING THE SUBTRACTION OF OVERLAPPING DIGITS.
C
        DO 120 J=1,DP
          POS=P+1-J
          ITOT=X(POS)-Y(POS+NORM)-IBRW
          IF (ITOT.GE.0) GOTO 100
            IBRW=1
            ITOT=ITOT+10000
          GOTO 110
100     IBRW=0
110     Z(POS)=ITOT
120     CONTINUE
C
C    IF A BORROW WAS NOT GENERATED ON THE LAST SUBTRACTION ASSIGN
C    Z THE CORRECT VALUE AND RETURN.
C
        IF (IBRW.NE.0) GOTO 130
          CALL ASSIGN(Z,X,POS-1)
          RETURN
C
C    CONTINUE SUBTRACTION UNTIL A ZERO BORROW IS GENERATED.
C
130     POS=POS-1
        ITOT=X(POS)-1
        IF (ITOT.GE.0) GOTO 140
            Z(POS)=9999
            GOTO 130
140     CALL ASSIGN(Z,X,POS)
        Z(POS)=ITOT
```

```
C
C    IF A DIGIT HAS NOT BEEN LOSS TO SUBTRACTION RETURN
C
        IF (POS.GT.3) RETURN
        IF (ITOT.GT.0) RETURN
            IP1=P-1
C
C    NORMALIZE THE NUMBER Z AND FILL THE LOST DIGIT WITH THE
C    GAURD DIGIT.
C
            DO 150 J=3,IP1
                Z(J)=Z(J+1)
150         CONTINUE
          Z(P)=G
          Z(2)=Z(2)-1
          CALL VNORM(Z,P)
      RETURN
      END
      SUBROUTINE VNORM(Z,P)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C    THE SUBROUTINE VNORM(Z,P) NORMALIZES THE MP NUMBER Z.             C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION Z(P)
C
C    FIND THE NUMBER OF LEADING ZEROS.
C
      DO 10 J=3,P
        IF (Z(J).NE.0) GOTO 20
10    CONTINUE
C
C    THE NUMBER Z IS ZERO. SET THE SIGN DIGIT TO INDICATE Z IS
C    ZERO AND RETURN.
C
50    Z(1)=0
      RETURN
20    SHIFT=J-3
C
C    IF THERE ARE NO LEADING ZEROS RETURN.
C
      IF (SHIFT.EQ.0) RETURN
C
C    ADJUST THE EXPONENT OF Z.
C
        Z(2)=Z(2)-SHIFT
C
C   SHIFT OUT THE LEADING ZEROS.
C
        PS=P-SHIFT
        DO 30 J=3,PS
          Z(J)=Z(J+SHIFT)
30      CONTINUE
        PS=PS+1
C
C   FILL WITH ZEROS.
C
        DO 40 J=PS,P
          Z(J)=0
40      CONTINUE
      RETURN
      END
```

9. APPENDIX B

```
      SUBROUTINE PBETA(P,Q,PROB,PPOINT,IER)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                        C
C     THE SUBROUTINE PBETA(P,Q,PROB,PPOINT,IER) IS A DOUBLE PRECISION    C
C     SUBROUTINE FOR FINDING PERCENTAGE POINTS OF THE BETA               C
C     DISTRIBUTION. AN ITERATIVE PROCEDURE IS USED TO FIND THE           C
C     PERCENATGE POINT SUCH THAT PROB=I(P,Q;PPOINT), WHERE I(P,Q;X)      C
C     IS THE INCOMPLETE BETA FUNCTION WITH PARAMETERS P AND Q. LET       C
C     Z(X(N))=PROB-I(P,Q;X(N))                                           C
C     F(X(N)) BE THE BETA DENSITY FUNCTION EVALUATED AT X(N)             C
C     C1(X(N))=1                                                         C
C     C2(X(N))= -((P-1)/X(N)-(Q-1)/(1-X(N)))                             C
C     C3(X(N))=2*(C2(X(N))**2 + (P-1)/(X(N)**2) + (Q-1)/((1-X(N))**2)    C
C                                                                        C
C     THE ITERATIVE EQUATION IS AS FOLLOWS:                              C
C                                                                        C
C     X(N+1)=X(N) + (Z(X(N))/F(X(N))) +                                  C
C            C2(X(N))/2 * (Z(X(N))/F(X(N)))**2 +                         C
C            C3(X(N))/6 * (Z(X(N))/F(X(N)))**3                           C
C                                                                        C
C    X(0)=.5   IF THE VALUE X(N+1) GENERATED FROM X(N) LIES OUTSIDE      C
C    OF THE INTERVAL (0,1), X(N+1) IS FOUND USING BISECTION. THE         C
C    METHOD GIVEN ABOVE HAS CONVERGENCE OF ORDER 4. THE VARIABLE         C
C    IER IS AN INTEGER AND HAS THE VALUE 0 IF PROB IS A VALID            C
C    PROBABLITY. OTHERWISE, IER HAS THE VALUE 1.                         C
C                                                                        C
C    INORDER TO TRY TO AVOID AN EXPONENT OVERFLOW WHEN COMPUTING         C
C    THE BETA DENSITY, THE LOG OF THE DENSITY IS COMPUTED.               C
C    THE BETA DENSITY F(X) IS GIVEN BELOW.                               C
C                                                                        C
C    F(X)=(GAMMA(P+Q)/(GAMMA(P)*GAMMA(Q))) * X**(P-1) * (1-X)**(Q-1)     C
C                                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT REAL*8 (A-Z)
      INTEGER IER
      IER=0
      LOW=0.D0
      HIGH=1.D0
      XN=.5D0
C
C     IF PROB>1 OR PROB<0 SET IER=1 AND RETURN
C
      IF(PROB.LT.0.D0.OR.PROB.GT.1.D0) GOTO 40
      IF(PROB.GT.0.D0) GOTO 5
      PPOINT=0.D0
      RETURN
5     IF(PROB.LT.1.D0) GOTO 10
```

```
      PPOINT=1.D0
      RETURN
C
C    FIND FXN=F(X(N))
C
10    FXN=DLGAMA(P+Q)-DLGAMA(P)-DLGAMA(Q)+(P-1)*DLOG(XN)+
     1 (Q-1)*DLOG(1.D0-XN)
      FXN=DEXP(FXN)
C
C    FIND C2XN=C2(X(N)), ZXN=Z(X(N)), ZDFXN=(Z(X(N))/F(X(N)), AND
C    C3XN=C3(X(N))
C
      PSIXN=-((P-1.D0)/XN-(Q-1.D0)/(1.D0-XN))
      CALL IBETA1(P,Q,XN,16,PRXN)
      ZXN=PROB-PRXN
      ZDFXN=ZXN/FXN
      C2XN=PSIXN
      C3XN=2.D0*(PSIXN**2)+(P-1.D0)/(XN**2)+(Q-1.D0)/((1-XN)**2)
C
C    STORE X(N) IN XNL
C
      XNL=XN
C
C    FIND X(N+1)
C
      XN=XN+ZDFXN+C2XN*(ZDFXN**2)/2+C3XN*(ZDFXN**3)/6
C
C    CHECK THAT X(N+1) LIES IN THE INTERVAL (0,1)
C
      IF(XN.LT.1.D0.AND.XN.GT.0.D0) GOTO 30
C
C    THE ITERATIVE METHOD IS DIVERGING.  USE BISECTION TO FIND
C    X(N+1).
C
      IF(PRXN.LT.PROB) GOTO 20
      IF (XNL.LE.HIGH) HIGH=XNL
      XN=LOW+(HIGH-LOW)/2
      GOTO 10
20    IF(XNL.GT.LOW) LOW=XNL
      XN=LOW+(HIGH-LOW)/2
      GOTO 10
C
C    CHECK THAT THE RELATIVE ERROR OF THE I(P,Q;X(N+1)) TO PROB
C    IS SUFFICIENTLY SMALL
C
30    IF(DABS(ZXN/PROB).GT..1D-12) GOTO 10
      PPOINT=XN
      RETURN
```

```
40      IER=1
        RETURN
        END
```