Interval Computation of Gamma Probabilities and their Percentile Points

A new method for computing the gamma cumulative distribution functions and their percentile points is presented in this paper. This method uses two continued fractions for computation, one for the incomplete gamma function and the other for the complement of the incomplete gamma function. An improved interval method for computation and implementation is introduced and C++ language classes is used. This is a self-validated computation. We developed programming techniques to speed up the increment in the iterative loops for finding the inverse of the gamma cumulative distribution function for a given probability. In fact, the percentile points can be considered random gamma variates if a uniform random number generator is used to generate the probabilities over the interval \([0, 1)\). The entire computation only involves two simple algebraic functions. There is no use of transcendental functions, auxiliary functions, power series, or Newton’s method in the computation. Therefore, one can expect it is easy to implement.

KEY WORDS: Continued fraction; Gamma distribution function; C++ programming language; Chi-square distribution function; incomplete gamma function; C++ class.

1. INTRODUCTION

The gamma distribution function is characterized by two parameters, \(\alpha\) and \(\beta\). A random variable \(X\) that has a probability density function of the following form is said to have a gamma distribution:

\[
g_x(x) = \begin{cases} 
\frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-x/\beta}, & \text{for } 0 \leq x < \infty, \text{ and } \alpha, \beta > 0, \\
0, & \text{otherwise}.
\end{cases}
\] (1.1)
where \( \Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} \, dx \).

Note that the density function of the gamma distribution given in (1.1) is the Chi-square distribution with \( r \) degrees of freedom, if \( \beta = 2 \) and \( \alpha = r/2 \), where \( r \) is a positive integer. The density function of the gamma distribution becomes an exponential distribution, if \( \alpha = \beta = 1 \).

The parameter \( \alpha \) plays the role of a scale parameter. This is apparent if the random variable \( X \) is gamma with parameters \( \alpha \) and \( \beta \), then the random variable \( \beta X \) is gamma with parameters \( \alpha \) and 1 with its probability density function given by

\[
g_{\beta \alpha}(x) = \begin{cases} \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x}, & \text{for } 0 \leq x < \infty \text{ and } \alpha, \beta > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{1.2} 
\]

We will study the gamma probability density function given in (1.2) and its cumulative distribution function:

\[
G(x; \alpha) = \frac{1}{\Gamma(\alpha)} \int_0^x t^{\alpha-1} e^{-t} \, dt, \quad \text{for } 0 \leq x < \infty, \text{ and } \alpha > 0, \tag{1.3}
\]

where \( \Gamma(\alpha) \) is called the complete gamma function.

Let

\[
\gamma(x; \alpha) = \int_0^x t^{\alpha-1} e^{-t} \, dt \tag{1.4}
\]

which is often called the incomplete gamma function.

In this paper, we develop a new procedure that uses two continued fractions to compute gamma probabilities and their percentile points. It is easy to implement, accurate, and fast. Section 2 contains a brief discussion of available tables and the most recent works in the area. In Section 3, we will derive the main procedure and describe the implementation of our method.
An improved interval method is defined in Section 4. Some special implementation techniques and testing results are presented in Section 5. Conclusions are given in the last section.

2. AVAILABLE TABLES AND RECENT WORK

Currently available tables, such as Standard Mathematical Tables by Selby (1973), give very rough values for the gamma probabilities and their percentile points. They contain only 5 digits after the decimal point and up to only 30 degrees of freedom. For any greater number of degrees of freedom, users are required to use a standard normal approximation. However, standard normal probability tables are not accurate and their approximations contain some unspecified errors. Even though tables in the Handbook of Mathematical Functions by Abramowitz and Stegun (1972) for the Chi-square distribution are better than Selby’s tables, they are still insufficiently accurate for many applications. They can only provide 4 to 6 digits after the decimal point and in most cases the last digit is not accurate. Therefore, an accurate computation of the gamma distribution is essential.

Usually the computation of the gamma function, \( \Gamma(\alpha) \), is divided into two components, the incomplete gamma function and the complement of the incomplete gamma function:

\[
\gamma(x;\alpha) = \int_0^x t^{\alpha-1} e^{-t} \, dt, \quad \text{and} \quad \Gamma(x;\alpha) = \int_x^\infty t^{\alpha-1} e^{-t} \, dt \tag{2.1}
\]

and we have

\[
\Gamma(\alpha) = \gamma(x;\alpha) + \Gamma(x;\alpha). \tag{2.2}
\]

Hence, the gamma distribution function given in (1.3) can be written as

\[
G(x;\alpha) = \frac{\gamma(x;\alpha)}{\gamma(x;\alpha) + \Gamma(x;\alpha)}, \quad \text{for } 0 \leq x < \infty, \text{ and } \alpha > 0.
\]
Accurate computation of the *complement of the incomplete gamma function*, $\Gamma(x; \alpha)$, is not an easy task. Computer scientists, mathematicians, and statisticians have worked for more than half a century approximating $\Gamma(x; \alpha)$. However, the results are still not satisfactory.

In the early 1970’s, Dingle (1973) and Olver (1974) developed an asymptotic expansion for the gamma function.

$$\Gamma(x; \alpha) = e^{-x} - x^{\alpha-1} \left[ 1 + \frac{\alpha - 1}{x} + \frac{\alpha - 1}{x^2} + \ldots \right].$$

These approximations for $\alpha$ are $o(x)$ only. For large values of $x$ and $\alpha$, Temme (1975) obtained a better approximation in terms of the *complementary error function*:

$$G(x; \alpha) = \frac{\gamma(x; \alpha)}{\Gamma(\alpha)} = \frac{\text{erfc}(\alpha^{1/2} \zeta)}{2} - R(x; \alpha)$$

and

$$Q(x; \alpha) = \frac{\Gamma(x; \alpha)}{\Gamma(\alpha)} = \frac{\text{erfc}(\alpha^{1/2} \zeta)}{2} + R(x; \alpha)$$

where $\zeta$ is a function of $(x/\alpha)$,

$$R(x; \alpha) = \frac{e^{-\phi(\lambda)}}{2\pi i} \sum_{k=0}^{\infty} c_k(\lambda) \Gamma(k + 1/2)(\alpha / 2)^{-k-1/2},$$

$\phi(t) = t - 1 - \ln t$, and coefficient $c_k(\lambda)$ can be found in Temme (1975).

Moreover, Temme (1979) extended his *uniform asymptotic expansions* to include complex variables and estimations for the remainder. Fettis (1979) used an asymptotic expansion to estimate the upper percentage points of the *Chi-square distribution*. These four mathematicians characterized important mathematical properties and made significant contributions in the study of the *gamma function*.

In 1979, Gautschi (1979) proposed a computational procedure for the *incomplete gamma function* $\gamma(x; \alpha)$ and its complementary function, $\Gamma(x; \alpha)$. Gautschi evaluated
\[ \gamma^*(x; \alpha) = \frac{x^{-\alpha}}{\Gamma(\alpha)} \gamma(x; \alpha), \]

by a power series to obtain

\[ \gamma^*(x; \alpha) = \frac{M(\alpha, \alpha + 1; -x)}{\Gamma(\alpha + 1)}, \]

where \( M(\alpha, \alpha + 1; -x) \) is a power series, and \( \Gamma(x; \alpha) = \Gamma(\alpha)[1 - x^{\alpha} \gamma^*(x; \alpha)]. \)

Gautschi has made a substantial contribution in the computation of gamma probabilities. But the implementation of the procedure is not easy. Evaluation of the power series and piece wise approximations for an accurate result are time consuming. For \( \Gamma(x; \alpha) \), he used a combination of methods including direct evaluation based partly on power series, recursive computation, and the classical continued fraction for various regions of the variable \( x \). All of the above require evaluation of transcendental functions and power series that can lead to some inaccurate results due to the generation of truncation errors every time a series is used.

Recently, two researchers Didonato and Morris (1986) of the Naval Surface Warfare Center, used three approximations to perform computations for three different regions of \( x \) values.

(1) \[ G(x; \alpha) = \frac{\gamma(x; \alpha)}{\Gamma(\alpha)} = \frac{R(x; \alpha)}{\alpha} \left[ 1 + \sum_{n=1}^{\infty} \frac{x^n}{(\alpha + 1)(\alpha + 2)\ldots(\alpha + n)} \right]. \]

(2) \[ Q(x; \alpha) = \frac{\Gamma(x; \alpha)}{\Gamma(\alpha)} = \frac{R(x; \alpha)}{x} \left[ 1 + \sum_{n=1}^{N-1} \frac{(\alpha + 1)(\alpha + 2)\ldots(\alpha + n)}{x^n} + \frac{\theta_N \alpha x}{x^N} \right], \]

where \( \theta_N = e^x x^{N+1-\alpha} \int_x^\infty t^{\alpha-1} e^{-t} dt, \) for \( N > 1 \),

and \( R(x; \alpha) = \frac{e^{-x} x^\alpha}{\Gamma(\alpha)}, \) for \( \alpha > 0, x \geq 0. \)

(3) \[ Q(x; \alpha) = R(x; \alpha) \left[ \frac{1}{x + \frac{1}{x + \frac{1}{x + \ldots}}} \right], \] a continued fraction.
(4) \( Q(x; \alpha) = e^{-x} \sum_{n=0}^{\alpha-1} \frac{x^n}{n!}, \alpha = 1, 2, \ldots \)

and \( Q(x; \alpha) = \text{erf} \sqrt{x} + \frac{e^{-x}}{\sqrt{\pi}} \sum_{n=1}^{i} \frac{x^n}{(1-1/2)(2-1/2)\ldots(n-1/2)}, \) for \( \alpha = i + \frac{1}{2} (i=1, 2, \ldots). \)

When \( i \) is finite, (4) is used for \( \alpha \leq x < x_0 \), where \( x_0 \) is the smallest value that

\[
\theta_N = e^x x^{N+\alpha} \int_x^{\infty} t^{\alpha-N-1} e^{-t} dt, \text{ for } N > 1,
\]

is applied, and \( \alpha = n/2 \) for integer \( n \geq 2 \). Otherwise, they split the region of \( x \) into three cases for \( 1 \leq \alpha < \text{BIG} \):

If \( x \leq \max [\alpha, \ln 10] \), then (1) is applied; else

if \( x > x_0 \), then (2) is used, and

if \( x < x_0 \), then (3) is used.

Their procedure represents considerable progress in the computation of the incomplete gamma function. Nevertheless, the implementation of their formulae and procedure are not simple and the computation of a summation or integration takes significant time for obtaining accurate values. It can only apply the continued fraction for a small region of \( x \) in the computation. The other problem is the use of transcendental functions, power series, auxiliary functions, and other infinite series, which can induce truncation errors in the course of computation.

Wang and Kennedy (1994) developed some methods to compute probabilities for some selected central and non-central univariate probability functions. They used Taylor series, transcendental functions, and power series for computations. In their conclusion they wrote, “When a failure occurred, an excessively large interval resulted. This served to notify of failure.
The methods are not completely fail safe, because the results are not valid if floating-point underflow or overflow occurs. But underflow or overflow can be detected, these methods do provide a large measure of dependability.” The overflow and underflow are crucial problems in the computation of gamma probabilities and their percentile points.

Again, Gautschi (1999) developed a recursive procedure to calculate incomplete Gamma functions. From his Table 1 (Gautschi 1999, p. 106) the maximum relative errors are from 0.9599D-13 to 0.1178D-11. In some cases, the errors become significantly larger or can cause overflow. It must take logarithms and then exponentiation for computation.

To overcome the above limitations and problems, one first needs to develop a new procedure that does not use transcendental functions, auxiliary functions, power series, and other infinite series for computation. Below, we propose a new method that uses two continued fractions to compute $\Gamma(x; \alpha)$ and $\gamma(x; \alpha)$ for entire region of $0 \leq x < \infty$.

3. THE MAIN PROCEDURE

To compute the gamma probabilities and their percentile points, we propose the following:

(1) Use of a continued fraction, originally due to Legendre, is given in Abramowitz and Stegun [1972, p.263, eq.6.5.31] and in Khooanskii (1963, p144), for computation of $\Gamma(x; \alpha)$:

$$e^x \Gamma(x; \alpha) = e^x \int_x^\infty t^{\alpha-1} e^{-t} dt = x^{\alpha} \left[ \frac{1 - \alpha}{x + 1} \frac{1}{x + 1} \frac{2 - \alpha}{x + 1} \frac{2}{x + 1} \cdots \right].$$

Hence, $\Gamma(x; \alpha) = e^{-x} x^{\alpha} \left[ \frac{1 - \alpha}{x + 1} \frac{1}{x + 1} \frac{2 - \alpha}{x + 1} \frac{2}{x + 1} \cdots \right]$, for any $x \geq 0$. (3.1)
(2) Again, we use another continued fraction expansion, from Khooanskii (1963, p150), for computing of the incomplete gamma function \( \gamma(x; \alpha) \) for \( x \geq 0 \).

\[
x^{-\alpha} e^x \gamma(x; \alpha) = x^{-\alpha} e^x \int_0^x t^{\alpha-1} e^{-t} dt
\]

\[
= \left[ \frac{1}{\alpha} \frac{\alpha x}{\alpha - (1 + \alpha + x)} - \frac{(1 + \alpha)x}{(2 + \alpha + x)} - \frac{(2 + \alpha)x}{(3 + \alpha + x)} - \cdots \right].
\]

Hence,

\[
\gamma(x; \alpha) = e^{-x} x^\alpha \left[ \frac{1}{\alpha} \frac{\alpha x}{\alpha - (1 + \alpha + x)} - \frac{(1 + \alpha)x}{(2 + \alpha + x)} - \frac{(2 + \alpha)x}{(3 + \alpha + x)} - \cdots \right]. \tag{3.2}
\]

(3) Let

\[
A(x; \alpha) = \left[ \frac{1}{x+1} \frac{1-\alpha}{x} \frac{1}{x+1} \frac{2-\alpha}{x} \frac{2}{x+1} \frac{2}{x+1} \cdots \right]. \tag{3.3}
\]

and

\[
B(x; \alpha) = \left[ \frac{1}{\alpha} \frac{\alpha x}{\alpha - (1 + \alpha + x)} - \frac{(1 + \alpha)x}{(2 + \alpha + x)} - \frac{(2 + \alpha)x}{(3 + \alpha + x)} - \cdots \right]. \tag{3.4}
\]

For the gamma function \( \Gamma(\alpha) = \gamma(x; \alpha) + \Gamma(x; \alpha) \), after cancellation of the common factor, \( e^{-x} x^\alpha \), we have the **gamma cumulative distribution function** as follows:

\[
G(x; \alpha) = \frac{\gamma(x; \alpha)}{\Gamma(\alpha)} = \frac{\gamma(x; \alpha)}{\Gamma(x; \alpha) + \gamma(x; \alpha)}
\]

\[
= \frac{B(x; \alpha)}{A(x; \alpha) + B(x; \alpha)}, \tag{3.5}
\]

for \( 0 \leq x < \infty \), and \( \alpha > 0 \).

Note that the mathematical functions \( A(x; \alpha) \) and \( B(x; \alpha) \) in equations (3.3), (3.4), and (3.5) are rational functions that contain no transcendental factors, power series, partitioning into sub-regions, or piece-wise approximations. The computation of \( A(x; \alpha) \) and \( B(x; \alpha) \) only
involves a finite number of additions, subtractions, multiplications, and divisions in terms of $x$ and $\alpha$. Modern computer systems can compute them to a very accurate level, usually up to the second to the last digit. To compute functions $A(x; \alpha)$ and $B(x; \alpha)$, we will develop recursive functions for the continued fractions given in (3.3) and (3.4). We apply the theory of continued fractions and matrix representation to convert a continued fraction into its equivalent recursive functions. This method is simple and easy. However in the actual computation, we will implement the recursive functions as an iterative loop, instead of recursive function calls. The iterative method saves time in computation.

(4) To compute $A(x; \alpha)$, let $r_n/s_n$ be the $n$th convergent of the continued fraction given in (3.3).

We have:

$$r_1 = 1, \ s_1 = x; \quad r_2 = 1, \ s_2 = 1 - \alpha + x.$$

For odd number terms, the 3rd, the 5th, . . . terms:

$$r_{2n+1} = x r_{2n} + n r_{2n-1}, \quad s_{2n+1} = x s_{2n} + n s_{2n-1}, \quad \text{for } n = 1, 2, 3, \ldots.$$

For even number terms, the 4th, the 6th, . . . terms:

$$r_{2n+4} = r_{2n+3} + (n + 2 - \alpha) r_{2n+2}, \quad s_{2n+4} = s_{2n+3} + (n + 2 - \alpha) s_{2n+2}, \quad \text{for } n = 0, 1, 2, \ldots.$$

Therefore, $A(x; \alpha) = \lim_{n \to \infty} \frac{r_n}{s_n}$.

(5) To compute $B(x; \alpha)$, let $p_n/q_n$ be the $n$th convergent of the continued fraction given in (3.4).

We have:

$$p_1 = 1, \quad q_1 = \alpha, \quad p_2 = 1 + \alpha + x, \quad q_2 = \alpha(1 + \alpha).$$

The 3rd, 4th, . . . terms are
\[ p_n = (n - 1 + \alpha + x) p_{n-1} - (n - 2 + \alpha) x p_{n-2}, \quad \text{for } n = 3, 4, \ldots \]

\[ q_n = (n - 1 + \alpha + x) q_{n-1} - (n - 2 + \alpha) x q_{n-2}, \quad \text{for } n = 3, 4, \ldots \]

Therefore, \( B(x; \alpha) = \lim_{n \to \infty} \frac{p_n}{q_n}. \)

(6) To compute the inverse \( x \) for given parameter \( \alpha \) and probability \( G_0 \), start with a guess value for \( x \) and compute \( G(x; \alpha) \). If \( (G_0 - G(x; \alpha)) > 4 \ast \text{eps} \), where \( \text{eps} \) is the machine epsilon, increase \( x \) by a unit of increment, \( (G_0 - G) \ast \text{step} \) (the value of \( \text{step} \) is given in Section 5, equation (5.1)), and repeat this process until \( G(x; \alpha) \) converges to \( G_0 \).

This procedure not only provides a straightforward computation, but also a more accurate result and quicker convergence. The most difficult part of numerical computation is determining the accuracy of the approximation. Therefore, we developed an improved interval method for computation that is implemented in C++ classes. The results of the computation will provide verification of the magnitude of the absolute error.

4. AN IMPROVED INTERVAL METHOD

In the last four decades, researchers introduced a new idea for performing computer arithmetic with compact intervals of real numbers instead of the direct use of scalars by Moore (1966, 1979), and Corliss and Rall (1987). This method is called interval computation. We revise ordinary interval arithmetic to dyadic interval arithmetic that replaces the original interval real endpoints with dyadic number (Kelly 1955) endpoints. In ordinary interval arithmetic, the real endpoints usually cannot be represented by a computer system exactly. In dyadic interval arithmetic, the dyadic number endpoints are binary numbers that always can be represented exactly by a computer system theoretically. For those dyadic numbers that can be exactly
represented by a given computer system, the Ada programming language (Watt, Wichmann, and Findlay 1987) calls them *model numbers*, with respect to a given computer. Using interval computation, we are able to guard against the accumulation of computational rounding errors, which are created by all floating-point number operations.

For any real numbers \( x \) and \( y \), there exist basic model intervals \( X = [a_1, b_1] \) and \( Y = [a_2, b_2] \) such that \( x \in [a_1, b_1] \) and \( y \in [a_2, b_2] \) where \( a_1, b_1, a_2, \) and \( b_2 \) are model numbers. For a real number \( c \) that is a model number, we consider a degenerate case \( c \in [c, c] \). To implement dyadic interval arithmetic, let \( \text{bin}(x) \) be the binary representation of real number \( x \), then we take \( a_1 = \text{bin}(x) - 2(\epsilon_{\text{p}}) \), and \( b_1 = \text{bin}(x) + 2(\epsilon_{\text{p}}) \), where the \( \epsilon_{\text{p}} \) is the largest number such that \( x + \epsilon_{\text{p}} = x \) in the given computer. For example: \( x = 1.0 \), the \( \epsilon_{\text{p}} \) is the *machine epsilon*. The usual interval arithmetic operations, addition +, subtraction −, multiplication *, and division /, are studied and then improved to meet model number requirements:

**Addition**

\[
x + y \in [a_1, b_1] + [a_2, b_2] = [a_1 + a_2, b_1 + b_2].
\]

**Subtraction**

\[
x - y \in [a_1, b_1] - [a_2, b_2] = [a_1 - a_2, b_1 - b_2].
\]

**Multiplication**

\[
x \ast y \in \{ \min(a_1a_2, a_1b_2, a_2b_1, b_1b_2), \max(a_1a_2, a_1b_2, a_2b_1, b_1b_2) \}.
\]

**Division**

For \( 1/y \in [1/b_2, 1/a_2] \) and \( 0 \not\in [1/b_2, 1/a_2] \), then we define

\[
x/y \in [\min(a_1/b_2, a_1/a_2, b_1/b_2, b_1/a_2), \max(a_1/b_2, a_1/a_2, b_1/b_2, b_1/a_2)].
\]

The deficiency of interval arithmetic is the use of real numbers for the lower bound and upper bound for computation. In general, a real number cannot be represented exactly by the floating-point number format. A better way to perform interval computation and reduce rounding-error
is: (1) to create the smallest closed interval, containing the given real number, with model numbers for the lower and upper bounds, and (2) to develop bit-to-bit computation over the mantissa, the internal representation of model numbers.

Since the product of two model numbers or the inverse of a model number might not be a model number, we round the result outward to the nearest model numbers. Let \( m(x) \) be the greatest model number less than \( x \) for the lower bound and \( m(y) \) be the smallest model number greater than \( y \) for the upper bound. Therefore, we redefine:

**Multiplication**

\[
x \cdot y \in [m(\min(a_1a_2, a_1b_2, b_1a_2, b_1b_2)), m(\max(a_1a_2, a_1b_2, b_1a_2, b_1b_2))]. \tag{4.5}
\]

**Division**

For \( 1/y \in [1/b_2, 1/a_2] \), and \( 0 \notin [1/b_2, 1/a_2] \), then we define

\[
x/y \in [m(\min(a_1/b_2, a_1/a_2, b_1/b_2, b_1/a_2)), m(\max(a_1/b_2, a_1/a_2, b_1/b_2, b_1/a_2))]. \tag{4.6}
\]

In 1966, Moore proved an important theorem that supports interval computation, and since then interval computation has become a new and growing branch of applied mathematics. The theorem is called the fundamental theorem of interval computation [Moore 1966]. We state the theorem here to support our work:

**Fundamental Theorem of Interval Computation.** Let \( f(x_1, x_2, \ldots, x_n) \) be a rational function of \( n \) variables. Consider any sequence of arithmetic steps which serve to evaluate \( f \) with given arguments \( x_1, x_2, \ldots, x_n \). Suppose we replace the arguments \( x_i \) by the corresponding interval \( X_i(i = 1, 2, 3, \ldots, n) \) and replace the arithmetic steps in the sequence used to evaluate \( f \) by the corresponding interval arithmetic steps. The result will be an interval \( f(X_1, X_2, \ldots, X_n) \). This interval contains the value of \( f(x_1, x_2, \ldots, x_n) \) for all \( x_i \in X_i(i = 1, 2, \ldots, n) \).
To implement model interval computation, we define a C++ class to represent an interval for computation. Then, overloading functions for arithmetic operations, addition +, subtraction −, multiplication *, and division /, are defined as member functions in the class for model interval arithmetic. Therefore, the results of the sum, difference, product, and quotient of two real numbers are all within an interval respectively.

5. SAMPLE RESULTS AND PROGRAMMING TECHNIQUE

We have developed a program to implement model interval method and scalar computation in the C++ language to compute the gamma probabilities and their percentile points. The computation was done on a microcomputer equipped with a Pentium Pro 200 MHz processor, Microsoft Visual C++ compiler, and 64 bit floating-point numbers. Since Chi-square distribution tables are more popular and they usually provide more significant digits than gamma distribution tables, our sample computation focuses upon the special case of Chi-square probabilities, \( G(\chi^2; r) \), with \( r \) degrees of freedom, and its percentile points. They are the same as gamma probabilities \( G(x; \alpha) \), with \( x = \chi^2 / 2 \), and \( \alpha = r / 2 \). In order to verify our computation, we take \( \alpha = 5, 10, \ldots, 50 \) and \( x = 1.9701, 5.4354, \ldots, 38.954 \) respectively to compute the Chi-square probabilities. Actually, these values for \( r = 2\alpha \) degrees of freedom and respective \( \chi^2 \) values are chosen from the Chi-square distribution tables in the Handbook of Mathematical Functions. The results are expected to be in the neighborhood of 0.05. Similarly, we take \( \alpha = 5, 10, \ldots, 50 \) and \( x = 9.1535, 15.7052, \ldots, 62.171 \) respectively to compute the Chi-square probabilities. The target results are around 0.95. For the computation of the inverse for a given
value of $\alpha (= r / 2)$ and given probability values $G_{0.05}$ and $G_{0.95}$, we start with a guess for $x (= \chi^2 / 2)$ and to use an iterative loop to converge to the target value.

To compute the probability of $x$ with a given parameter $\alpha$, we do the following: (1) determine the lower bound $l$ and upper bound $u$ for the smallest interval containing $x$, and (2) because the cumulative distribution function is a monotone function, $G(l; \alpha) \leq G(x; \alpha) \leq G(u; \alpha)$ for $l \leq x \leq u$. We use $G(l; \alpha)$ and $G(u; \alpha)$ for the lower bound and upper bound of $G(x; \alpha)$. This way, the result is more accurate than using intervals in the computation. To compute an inverse, $\chi^2 (= 2x)$, for a given probability $G_0$ and given $r (= 2\alpha)$, we first compute the scalar value $\chi^2$, and then compute the interval containing $\chi^2$. From the results, we found almost all scalar values of $\chi^2$ fall in the middle of its interval. Next, we use the lower bound and the upper bound of $\chi^2$ to determine the lower bound and upper bound probabilities to see if these probabilities are in the neighborhood of $G_0(x; \alpha) (= G_0(\chi^2; r))$. For given $G_0$ and $r$, the computed inverse $\chi^2$, interval of $\chi^2$, and interval of $G_0$ are given in the Table 1 and Table 2: The computation time for all $\chi^2$ values and intervals of $\chi^2$ take only 0.150 seconds for Table 1 and 0.170 seconds for Table 2. The lower bounds and upper bounds of $G_0$ are computed separately, for each entry takes less than 1 millisecond for computation. The resulting interval provides verification of the magnitude of the absolute error for computation.

From Table 1, we obtained the maximum relative error of $\chi^2$ is $0.3183 \times 10^{-14}$ at $r = 90$ and the maximum relative error of $G_0$ is $0.5800 \times 10^{-13}$ at $r = 80$. From Table 2, the maximum relative error of $\chi^2$ is $0.4125 \times 10^{-14}$ at $r = 40$ and the maximum relative error of $G_0$ is $0.2211 \times 10^{-14}$ at $r = 80$. The computation time for all $\chi^2$ values and intervals of $\chi^2$ take only 0.150
seconds for Table 1 and 0.170 seconds for Table 2. The lower bounds and upper bounds of $G_0$ are computed separately, for each entry takes less than 1 millisecond for computation. The resulting interval provides verification of the magnitude of the absolute error for computation.

Table 1: Interval Results for Inverse and Prefixed Probability $G_0(\chi^2; r) = 0.05$

<table>
<thead>
<tr>
<th>$r$</th>
<th>$\chi^2$</th>
<th>Lower Bound of $\chi^2$</th>
<th>Upper Bound of $\chi^2$</th>
<th>Lower Bound of $G_0$</th>
<th>Upper Bound of $G_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3.940299136119058</td>
<td>3.940299136119054</td>
<td>3.940299136119061</td>
<td>0.0499999999999997</td>
<td>0.0500000000000001</td>
</tr>
<tr>
<td>20</td>
<td>10.85081139418259</td>
<td>10.85081139418258</td>
<td>10.85081139418261</td>
<td>0.0499999999999997</td>
<td>0.0500000000000002</td>
</tr>
<tr>
<td>30</td>
<td>18.49266098195347</td>
<td>18.49266098195344</td>
<td>18.49266098195350</td>
<td>0.0499999999999997</td>
<td>0.0500000000000000</td>
</tr>
<tr>
<td>40</td>
<td>26.50930319669315</td>
<td>26.50930319669310</td>
<td>26.50930319669315</td>
<td>0.0499999999999999</td>
<td>0.0500000000000001</td>
</tr>
<tr>
<td>50</td>
<td>34.76425168350172</td>
<td>34.76425168350170</td>
<td>34.76425168350175</td>
<td>0.0499999999999999</td>
<td>0.0500000000000002</td>
</tr>
<tr>
<td>60</td>
<td>43.18795845398981</td>
<td>43.18795845398975</td>
<td>43.18795845398985</td>
<td>0.0499999999999999</td>
<td>0.0500000000000003</td>
</tr>
<tr>
<td>70</td>
<td>51.73927804896290</td>
<td>51.73927804896290</td>
<td>51.73927804896295</td>
<td>0.0500000000000000</td>
<td>0.0500000000000001</td>
</tr>
<tr>
<td>80</td>
<td>60.39147838868949</td>
<td>60.39147838868946</td>
<td>60.39147838868955</td>
<td>0.0500000000000000</td>
<td>0.0500000000000002</td>
</tr>
<tr>
<td>90</td>
<td>69.12603042551541</td>
<td>69.12603042551530</td>
<td>69.12603042551552</td>
<td>0.0500000000000000</td>
<td>0.0500000000000003</td>
</tr>
<tr>
<td>100</td>
<td>77.92946516501706</td>
<td>77.92946516501695</td>
<td>77.92946516501718</td>
<td>0.0500000000000000</td>
<td>0.0500000000000004</td>
</tr>
</tbody>
</table>

Table 2: Interval Results for Inverse and Prefixed Probability $G_0(\chi^2; r) = 0.95$

<table>
<thead>
<tr>
<th>$r$</th>
<th>$\chi^2$</th>
<th>Lower Bound of $\chi^2$</th>
<th>Upper Bound of $\chi^2$</th>
<th>Lower Bound of $G_0$</th>
<th>Upper Bound of $G_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3.940299136119058</td>
<td>3.940299136119054</td>
<td>3.940299136119061</td>
<td>0.0499999999999997</td>
<td>0.0500000000000001</td>
</tr>
<tr>
<td>20</td>
<td>10.85081139418259</td>
<td>10.85081139418258</td>
<td>10.85081139418261</td>
<td>0.0499999999999999</td>
<td>0.0500000000000001</td>
</tr>
<tr>
<td>30</td>
<td>18.49266098195347</td>
<td>18.49266098195344</td>
<td>18.49266098195350</td>
<td>0.0499999999999999</td>
<td>0.0500000000000000</td>
</tr>
<tr>
<td>40</td>
<td>26.50930319669315</td>
<td>26.50930319669310</td>
<td>26.50930319669315</td>
<td>0.0499999999999999</td>
<td>0.0500000000000001</td>
</tr>
<tr>
<td>50</td>
<td>34.76425168350172</td>
<td>34.76425168350170</td>
<td>34.76425168350175</td>
<td>0.0499999999999999</td>
<td>0.0500000000000002</td>
</tr>
<tr>
<td>60</td>
<td>43.18795845398981</td>
<td>43.18795845398975</td>
<td>43.18795845398985</td>
<td>0.0499999999999999</td>
<td>0.0500000000000003</td>
</tr>
<tr>
<td>70</td>
<td>51.73927804896290</td>
<td>51.73927804896290</td>
<td>51.73927804896295</td>
<td>0.0500000000000000</td>
<td>0.0500000000000001</td>
</tr>
<tr>
<td>80</td>
<td>60.39147838868949</td>
<td>60.39147838868946</td>
<td>60.39147838868955</td>
<td>0.0500000000000000</td>
<td>0.0500000000000002</td>
</tr>
<tr>
<td>90</td>
<td>69.12603042551541</td>
<td>69.12603042551530</td>
<td>69.12603042551552</td>
<td>0.0500000000000000</td>
<td>0.0500000000000003</td>
</tr>
<tr>
<td>100</td>
<td>77.92946516501706</td>
<td>77.92946516501695</td>
<td>77.92946516501718</td>
<td>0.0500000000000000</td>
<td>0.0500000000000004</td>
</tr>
</tbody>
</table>

In the computation of $G(x; \alpha) = B(x; \alpha) / (A(x; \alpha) + B(x; \alpha))$ with given $\alpha$ and $G_0$ to find $x$ such that $G(x; \alpha) = G_0$, the given target value, we faced three problems: (1) the initial value of $x$ could be too small, it might take long time to converge, (2) if the increment of $x$ is too large, it might cross over the target value and run into an infinite loop, and (3) it is uncertain what preset maximum error should be allowed to terminate the loop. If the maximum error is too small, it
could become an infinite loop. If the allowed maximum error is too large, accuracy would be lost.

To resolve (1) and (2), we suggest that one use information about the difference between $G_0$ and $G$. More specifically, let

$$x \leftarrow x + (G_0 - G) \times \text{step}. \quad (5.1)$$

It turns out that for $\text{step} = 6$, $G$ converges to $G_0$ very quickly. For each $x$, it took less than 10 milliseconds to complete the computation. For a table it took less than one second to complete the computation. For problem (3), we used the condition ($|G_0 - G| \leq 4 \times \text{eps}$) to terminate the iteration, where $\text{eps}$ is the machine epsilon for double precision floating-point numbers. This condition works very well, and the error, $4 \times \text{eps}$, is still less than $10^{-15}$.

6. CONCLUSIONS

We have developed a procedure that uses two continued fractions to compute the gamma probabilities and their percentile points. These two continued fractions are the main factors of the incomplete gamma function and the complement of incomplete gamma function, respectively. Continued fractions are simple and easy to implement on a computer system. For this problem, it turns out that to compute the two rational functions $A(x; \alpha)$ and $B(x; \alpha)$, no auxiliary functions, piece wise approximations, transcendental functions, power series, integration, or Newton method were needed. Each computation merely uses a fixed number of addition, subtraction, multiplication, and division operations. It is a self-validated computation that provides very small absolute maximum error for a given result. In all test cases, we obtained 14 to 15 significant decimal digit accuracy form a 64 bit floating-point number machine. The
The computation time is very small that has recorded in the section 5. We have also built a mechanism that can reflect the accuracy of our computation. In fact, if a uniform random number generator is used to generate the probabilities over the interval [0, 1), the percentile points can be considered random gamma variates. For the entire project, there were some difficulties encountered in the implementation as we stated in the Section 5. Fortunately, we resolved those problems.

REFERENCES


