Efficient Algorithms for Higher-Order Derivatives of the Continued Erlang Delay Function

JORGE SÁ ESTEVES
University of Aveiro
Department of Mathematics
Campus de Santiago, 3810-193 Aveiro
PORTUGAL
saesteves@ua.pt

Abstract: In this paper we analyze the partial derivatives of any order of the continued Erlang delay function in the number of servers. Several properties with strong analytical relations between the high-order derivatives of Erlang's B and C functions are established. Using these relations, three algorithms are proposed for the numerical computation of the cited derivatives. For comparison purposes, it is also generalized a numerical method based on a quadrature procedure suggested by D. L. Jagerman [16]. All the computational methods are compared in terms of stability, efficiency and precision. Our study concludes that a recursive matrix relation presented in a previous work [10, 11], may be used for the establishment of a simple and reliable algorithm having the best performance considering the trade-off of the different criteria. Extensive computational results are presented and discussed. In the sequel, a conjecture about the strict convexity of the first derivative of Erlang delay function is presented and supported by numerical evidence.

Key–Words: Performance Evaluation, Queueing Systems, Erlang's B and C Formulas, Numerical Differentiation.

1 Introduction

The derivatives of Erlang C function are useful in the optimum design of stochastic service systems (queues or networks of queues). Actually, one of the most common models to support performance evaluation of those systems is the $M/M/x/\infty$ (Erlang C model). Moreover, the Erlang C formula plays an important role in approximations for more general systems.

Classical examples of such systems are telephone switching and cellular networks [7, 21], allocation of transport vehicles and of urban emergency units. Other examples are found in computer systems (jobs submitted to parallel processors, dynamic shared memory [20]), satellite communication systems (allocation of transmission bandwidth) and communications networks (teletraffic models) [13].

Recently, Erlang C model has been the subject of intensive study in the context of the dimensioning of call centers [12, 19, 14, 17]. A call center is a service network in which agents provide telephone-based services. Customers who seek these services are delayed in tele-queues. Worldwide, telephone-based services have been expanding dramatically. This has given rise to a huge growth industry — the (telephone) call center industry. In this context, the most common model to support workforce management of telephone call centers is the Erlang C system or its generalizations.

The most of the models consider non-linear performance functions related with Erlang's functions describing the behavior of the systems by using queueing theory. Examples of performance functions are the system throughput (total carried traffic by the system), grade of service, mean waiting time, average number of users in queue, etc.

The derivatives of performance functions are useful in the optimum design of those stochastic service systems. Non-linear programming problems encountered in optimizations of those systems can be solved by using the performance functions and their derivatives namely for Newton-Raphson and gradient type solutions. In fact, derivative-free optimization methods are not as well developed as gradient-based methods; current algorithms are effective only for small problems. Therefore, most algorithms for nonlinear optimization and nonlinear equations require knowledge of derivatives.

The usefulness of derivatives is not restricted to *algorithms* for solve equations and optimization problems. Modelers in areas such design optimization and economics are often interest in performing post-optimal *sensitivity analysis*, in which they determine the sensitivity of the optimum to small perturbations in the parameter or constraint values.

The obtainment of an efficient method, with good

accuracy, for calculating higher order derivatives of Erlang C function, beyond the interest in terms of numerical analysis, may be used for local approximations of the function and its derivatives by using a convenient Taylor polynomial. Those approximations are specially important for iterative algorithms in small neighborhoods of the solution. Indeed, those procedures could be much more efficient if the function and its derivatives are updated in each iteration using local approximations by using Taylor Theorem or other osculatory Hermite polynomial.

It is important to recognize that the Erlang C function is intimately related to the Erlang B function. In addition, Erlang's B and C functions may be easily related with very well known special functions, such as the confluent hypergeometric function, the incomplete gamma function and the chi-square probability function (see [1]). Therefore the algorithms for computing Erlang C function derivatives may have independent interest in other fields of applied Mathematics [9].

Two classical works on this subject are [2] and [3] which establish a numerical method for calculating derivatives until order two, by using a complex process based on the evaluation of some special functions related to the gamma function. Methods using numerical differentiation theory were approached in [22] but the obtained accuracy is relatively poor. The method derived from the Theorem 17 of [15], which leads to an accuracy of four significant figures for a wide range of the arguments, may also be included in this class of algorithms. A more accurate method of calculation of the derivatives of Erlang-B function using quadrature theory based on cardinal series, was proposed by D. L. Jagerman in a AT&T Bell Laboratories research report [16]. In Appendix A we propose a generalization of that method in order to compute directly the derivatives of Erlang C function of any order.

A method for calculating the derivatives of order n of the Erlang B function in the number of servers, which can be considered the natural generalization of the classical recursive algorithm for the calculation of the function, was developed by the author and presented in a previous paper [10]. This recursion, designated as generalized recursion, where both the number of servers and the order of the derivative intervene as parameters, was shown to have the important property that the propagated relative error associated with the calculation of the initial values always decreases (in a certain sense) and tends to a constant value, asymptotically. Extensive computation has shown that the proposed method is very accurate in an wide range of values of the arguments and compares favorably, in terms of efficiency, with the method proposed by D. L. Jagerman [16], excepting for very high values of the arguments where the situation is the inverse.

In the sequel, a second paper [11] shows that for high values of the arguments it is possible to obtain a significant improvement in the method efficiency, without jeopardizing the required precision, by defining a *reduced recursion* starting from a point closer to the desired value of the number of servers. The problem of estimating the initial values was by-passed, accepting the value zero for all initial values. It will be shown that this option does not jeopardize the accuracy of the method, since the absolute value of the relative error decreases rapidly during the recursive calculation.

The essential problem dealt with in paper [11] is focused on how to estimate the value of the initial point which allows to obtain the required precision. The proposed process is very efficient and inherently based on closed formulae avoiding iterative procedures. For short, the proposed method will be henceforth referred to as RR method (*Reduced Recursion Method*), by opposition to the general method proposed in the companion paper [10] henceforth referred to as CR method (*Complete Recursion Method*).

Following the above mentioned works on the Erlang B function derivatives, a method for calculating directly higher-order derivatives of Erlang C function, is developed in the present work.

2 The Erlang's B and C Formulas

The Erlang B and C formulas are true probability classics. Indeed, much of the theory was developed by A. K. Erlang and his colleagues prior to 1925 [6]. The subject has been extensively studied and applied by telecommunications engineers and mathematicians ever since. A nice introductory account, including some of the telecommunications subtleties, is provided by [8]. The Erlang B (or loss) formula gives the (steady-state) blocking probability in the Erlang loss model, i.e., in the $M/M/x/0$ model (see for example [8, pp. 5 and 79]):

$$B(a,x) \doteq \frac{a^x/x!}{\sum_{j=0}^{x} a^j/j!}, \quad x \in \mathbb{N}_0, \quad a \in \mathbb{R}^+. \quad (1)$$

The numerical studies regarding this formula are usually based on its analytical continuation, ascribed to R. Fortet [15]:

$$[B(a,x)]^{-1} = I(a,x) = a \int_0^{+\infty} e^{-az} (1+z)^x \, dz \quad (2)$$

which is valid for traffic offered $a \in \mathbb{R}^+$ and $x \in \mathbb{R}_0^+$ servers. The function $I(a,x)$ tends to be easier to analyze than $B(a,x)$ and may be named as the *reciprocal,* or the *inverse probability of blocking.*

A classical result is the following recursion obtained by partial integration of (2):

$$I(a,x+1) = \frac{x+1}{a}\, I(a,x) + 1, \quad x \in \mathbb{R}_0^+, \quad (3)$$

Since $B(a,0) = I_a(0) = 1$ for all $a \in \mathbb{R}^+$, $I(a,x)$ may be calculated by recursion (3) for any positive integer x. Actually, in [10] it is shown that (3) defines a very stable numerical recursion.

The Erlang C (or delay) formula gives the (steady-state) probability of delay (that an arrival must wait before beginning service) in the Erlang delay model. i.e., in the $M/M/x/\infty$ model (e.g., see pp. 7 and 91 of [8]):

$$C(a,x) \doteq \frac{\frac{a^x}{x!}\,\frac{x}{x-a}}{\sum_{j=0}^{x-1}\frac{a^j}{j!} + \frac{x}{x-a}}, \quad \text{for } 0 < a < x. \quad (4)$$

The Erlang C formula is intimately related to the Erlang B formula provided that $0 < a < x$. Indeed it is easy to show that:

$$C(a,x) = \frac{xB(a,x)}{x - a + aB(a,x)}, \quad 0 < a < x. \quad (5)$$

The *reciprocal* of $C(a,x)$ or the *inverse probability of waiting* may be defined as $J(a,x) \doteq 1/C(a,x)$. Using (5) and (3) it may be shown that

$$J(a,x) = I(a,x) - I(a,x-1). \quad (6)$$

An integral representation for the continued Erlang-C function may be obtained substituting relation (2) in (6):

$$[C(a,x)]^{-1} = J(a,x) = a\int_0^{+\infty} e^{-az}(1+z)^{x-1}\, z\, dz. \quad (7)$$

An integral representation for the continued Erlang-C function may be obtained substituting relation (2) in (6):

$$J(a,x) = a\int_0^{+\infty} e^{-az}(1+z)^{x-1}\, z\, dz \quad (8)$$

Relations (2) and (8) may be related if we introduce the *Confluent Hypergeometric or Kummer Function [1]* as

$$F(a,x,r) \doteq \frac{1}{\Gamma(r)}\int_0^{+\infty} e^{-az}(1+z)^{x-r+1} z^{r-1}\, dz, \quad (9)$$

where $\Gamma(r)$ is the well known Euler Gamma function:

$$\Gamma(r) \doteq \int_0^{+\infty} e^{-z}\, t^{z-1}\, dz. \quad (10)$$

Thence, we have:

$$I(a,x) = a\, F(a,x,0), \quad (11)$$
$$J(a,x) = a\, F(a,x,1). \quad (12)$$

The first order partial derivatives of $B(a,x)$ with respect to a and x are given by (see for example [15]):

$$B'_a(a,x) = \left[\frac{x}{a} - 1 + B(a,x)\right] B(a,x), \quad (13)$$
$$B'_x(a,x) = -[B(a,x)]^2\, I_1(a,x) \quad (14)$$

The derivatives of $B(a,x)$ and $C(a,x)$ of higher order in a may be computed from (13) and (5) and the associated numerical calculations is reduced to the evaluation of a rational expression in a, x and $B(a,x)$. In Section 4, methods for computing the derivatives $C_x^{(k)}(a,x),\ k = 0(1)n$ will be given. These methods are generalizations of those for the derivatives $B_x^{(k)}(a,x),\ k = 0(1)n$ proposed in [10] and [11].

3 Erlang B Function Derivatives

The main features of the method proposed in [10] are the starting point of this present work. In the sequel, we summarize the most important results from [10] and the corresponding notations used in this paper.

On account of the uniform convergence [5][pp. 44–45] of integrals (11) and (12), partial derivatives of $I(a,x)$ and $J(a,x)$ on variables a and x are given by simpler differentiation rules. Successive differentiation of (2) leads to:

$$I_k(a,x) = \frac{\partial^k I}{\partial\, x^k} = a\int_0^{+\infty} e^{-az}(1+z)^x \ln^k(1+z)\, dz, \quad (15)$$

and successive differentiation of (8) yields

$$J_k(a,x) = \frac{\partial^k J}{\partial\, x^k} = a\int_0^{+\infty} e^{-az}(1+z)^{x-1} z \ln^k(1+z)\, dz. \quad (16)$$

Successive differentiation of (3) with respect to x leads to the general recursive relation (valid for $k = 1, 2, \ldots$):

$$I_k(a,x+1) = \frac{x+1}{a}\, I_k(a,x) + \frac{k}{a}\, I_{k-1}(a,x). \quad (17)$$

Note that, if a convenient algorithm for calculating the values $I_k(a,x),\ k = 0(1)n$ is obtained, then it is possible to calculate all the derivatives $B_x^{(k)}(a,x),\ k = 0(1)n$ using Algorithm 2 (see Appendix A). The general recursive matrix relation for calculating derivatives of $B(a,x)$ in the number of servers x proposed in [10] is defined from (3) and (17). Also in [10] it

is shown this matrix scheme defines a very stable numerical recursion. For high values of the arguments (say $x > a > 100$), a significant improvement in efficiency may be obtained by considering a *reduced recursion* starting from a point closer to the desired value of x (see [11]). This can be attained without jeopardizing the required precision. For short, this method is referred to as RR method (*Reduced Recursion Method*), by opposition to the method proposed in [10] is referred to as CR method (*Complete Recursion Method*). The RR method has been established for derivatives until order two [11]. However, this method may be easily generalized for calculating higher order derivatives. Indeed, computational experiments show that the initial point used for calculating derivatives until order two may be used for calculating derivatives until order five with excellent precision (all the 15 figures are exact, except for some arguments greater than 10^5).

4 Erlang C Function Derivatives

Successive differentiation of (6) with respect to x leads to

$$J_k(a,x) = I_k(a,x) - I_k(a,x-1),\ k = 0,1,2,\ldots \tag{18}$$

Since we have a convenient algorithm to calculate the values $I_k(a,x)$, $k = 0(1)n$ relation (18) may lead to the computation of Erlang C function derivatives of any order of variable x. Unfortunately, for $x >> a$ we have $I_k(a,x) \approx I_k(a,x-1)$ and relation (18) is numerically unstable since the value calculated for $J_k(a,x)$ is affected by subtractive cancelation.

For this particular task, a reformulation of the problem of calculating $J_k(a,x), k = 0(1)n$ in terms of the quantities $I_k(a,x), k = 0(1)n$ avoids the difficulty. The following proposition s the first method for accomplish that reformulation.

Proposition 1 *For $k = 0,1,2,\ldots$ we have:*

$$J_k(a,x) = \sum_{j=1}^{+\infty} \frac{I_{k+j}(a,x-1)}{j!} = \sum_{j=1}^{+\infty} (-1)^{j+1} \frac{I_{k+j}(a,x)}{j!}.$$

Proof: The Taylor expansion of function $I_k(a,x)$ at point $\xi_x = x-1$ may be written as

$$I_k(a,x) = I_k(a,x-1) + \sum_{j=1}^{+\infty} \frac{I_{k+j}(a,x-1)}{j!}.$$

The first series follows using relation (18). Alternatively, from the Taylor expansion of $I_k(a,x-1)$ at point x we obtain

$$I_k(a,x-1) = I_k(a,x) + \sum_{j=1}^{+\infty} (-1)^j \frac{I_{k+j}(a,x)}{j!}.$$

Again, relation (18) gives the representation of $J_k(a,x)$ in the form of the second series. □

Proposition 1 may be used to compute $J_k(a,x)$ with arbitrary precision, since the truncation error of the partial summation of the series may be easily bounded. Indeed, since $J_k(a,x)$ is the sum of an alternated series, it follows that:

$$J_k(a,x) \approx \sum_{j=1}^{r} (-1)^{j+1} \frac{I_{k+j}(a,x)}{j!}, \tag{19}$$

with absolute error less than $I_{k+r+1}(a,x)/(r+1)!$. To estimate the magnitude of the decay of that bound of the error it is important to remember that (see Lemma 4 of Appendix A of [10]):

$$\lim_{x\to\infty} \frac{I_{k+1}(a,x)}{I_k(a,x)\,\ln(x)} = 1, \quad k = 0,1,2,\ldots \tag{20}$$

Therefore, for large x it is easy to see that

$$I_{k+r+1}(a,x)/(r+1)! \approx \frac{\ln^r(x)}{(r+1)!}\, I_{k+1}(a,x),$$

implying a slow convergence of the summation (19). Furthermore, the evaluation of the $I_{k+r}(a,x)$ for large r by recursion have an important computational cost. The conclusion is that the algorithm inspired in relation (19) is numerically stable but is rather inefficient.

The following proposition is another attempt in order to define a convenient method of computation of the $J_k(a,x)$ quantities.

Proposition 2 *For $a \in \mathbb{R}^+$ and $x \in [1,+\infty[$:*

$$\begin{aligned} J_0(a,x) &= \frac{x}{a} J_0(a,x-1) + \frac{1}{a} I_0(a,x-2) \\ J_k(a,x) &= \frac{k}{a} J_{k-1}(a,x-1) + \frac{x}{a} I_k(a,x-1) + \\ &+ \frac{1}{a} I_k(a,x-2), \quad k \in \mathbb{N}. \end{aligned}$$

Proof: Integration by parts applied to (8) leads to

$$\begin{aligned} J_0(a,x) &= \left[-e^{-az}(1+z)^{x-1} z\right]_0^{+\infty} + \\ &+ \int_0^{+\infty} e^{-az}(zx+1)(1+z)^{x-2}\, dz. \end{aligned}$$

Therefore it may be written

$$J_0(a,x) = \frac{x}{a}a\int_0^{+\infty} e^{-az}(1+z)^{x-2}z\,dz + \\ + \frac{1}{a}a\int_0^{+\infty} e^{-az}(1+z)^{x-2}\,dz,$$

thence the first relation holds. Successive differentiation with respect to variable x leads to the intended recursion for $J_k(a,x)$. □

Proposition (2) may have theoretical interest but its numerical stability it is an open question. Moreover there are some disadvantages related not only to the evaluation of initial values but also to the number of arithmetic operations involved.

Proposition 3 *For $a \in \mathbb{R}^+$, $x \in [1,+\infty[$ and $k \in \mathbb{N}$:*

$$J_0(a,x) = \frac{x-a}{a}\,I_0(a,x-1) + 1$$
$$J_k(a,x) = \frac{k}{a}I_{k-1}(a,x-1) + \frac{x-a}{a}\,I_k(a,x-1).$$

Proof: Since $I_0(a,x) = x\,I_0(a,x-1)/a + 1$, from (6) the first equality follows:

$$J_0(a,x) = \frac{x-a}{a}\,I_0(a,x-1) + 1.$$

Successive differentiation with respect to variable x leads to the second equality. □

Finally, Proposition (3) establishes a convenient computational method. Indeed, we only need to compute $I_k(a,x-1),\ k = 0(1)m$ using the matrix recursion defined in [10] and then apply Proposition 3 in order to compute $J_k(a,x),\ k = 0(1)m$. If a and x are greater than 100 the RR algorithm may be used to improve the computational efficiency. Algorithm 4 defines the computational procedure.

5 Computational Results

First note that $C(a,x)$ is a strictly decreasing function of x, therefore $C'_x(a,x) < 0$. Since $C(a,x)$ is a convex function in variable x (see [18]), $C''_x(a,x) \geq 0$. Numerical evidence, obtained by our computational experiments, leads to the conjecture $C'''_x(a,x) < 0$, implying that $C'_x(a,x)$ is also a convex function of x. However, the derivatives of higher order do not seem to preserve sign. Figures 1–4 show samples of graphic representations which illustrate these properties.

Algorithm 1 defines the CR method for the computation of high-order derivatives of Erlang-C function. The initial values are computed using Gauss-Laguerre quadrature (degree 15) as explained in [10].

Algorithm 1 CR Method for $C_x^{(n)}(a,x),\ n = 0(1)m$

Require: $a \in \mathbb{R}^+, x \in \mathbb{N}$ and $m \in \mathbb{N}_0$;
Ensure: $C_k = C_x^{(n)}(a,x)$ for $n = 0,1,\ldots,m$;
  Obtain initial values $I_n(a,0),\ n = 0(1)m$;
  for $j = 0$ to $x-1$ **do**
    for $k = m$ to 1 **do**
      $I_k \leftarrow (k \cdot I_{k-1} + j \cdot I_k)/a$;
    end for
    $I_0 \leftarrow 1 + j \cdot I_0/a$;
  end for
  $J_0 \leftarrow 1 + (x-a) \cdot I_0/a$;
  for $k = 1$ to m **do**
    $J_k \leftarrow (k \cdot I_{k-1} + (x-a) \cdot I_k)/a$
  end for
  Compute $C_x^{(n)},\ n = 0(1)m$ by Algorithm 2;

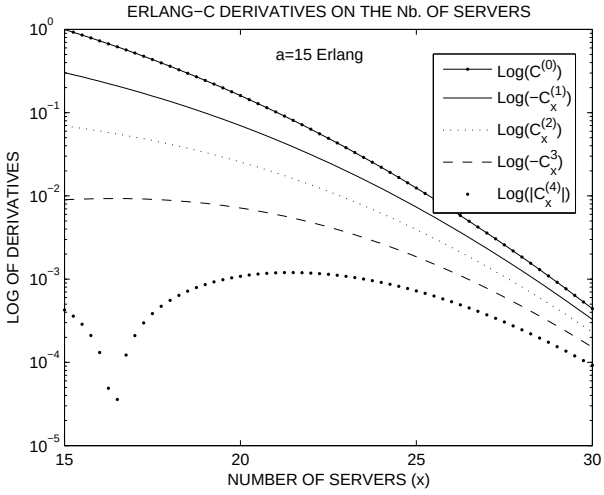

Figure 1: Graphics of $|C_x^{(k)}(15,x)|$ for $x \in [15,30]$ computed by CR Method (Algorithm 1).

The implementation of the correspondent RR method it is straightforward since the only modification is the point to start the recurrence calculations easily computed as indicated in [11]. The implementation of all the algorithms was performed using Turbo C compiler version 2.1. The measuring of processing times (durations of evaluations) was performed on a PC (Intel Core2 dual E6400 processor running at 2.13 GHz and 2 GB RAM) using MS Windows XP and time critical priority of the process. All the calculations were made in double precision, and tables are produced by writing automatically LaTeX code.

Extensive computational results have been obtained, enabling to compare the precision and efficiency of the RR method, the CR method and Jagerman algorithm (see Appendix B).

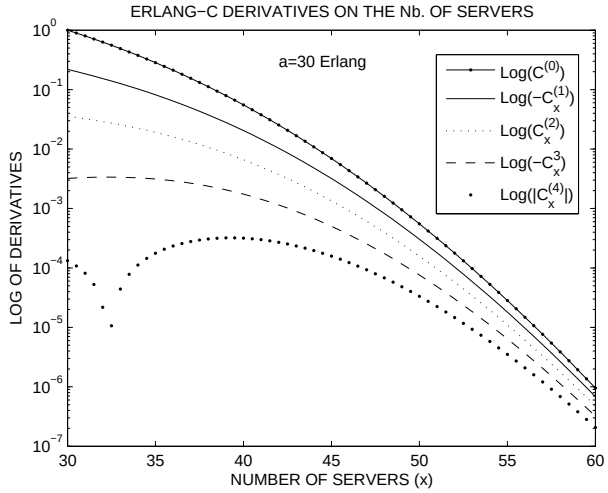

Figure 2: Graphics of $|C_x^{(k)}(30,x)|$ for $x \in [30,60]$ computed by CR Method (Algorithm 1).

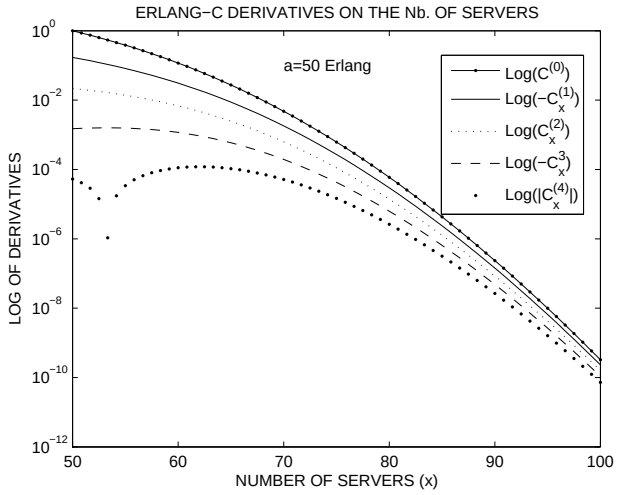

Figure 3: Graphics of $|C_x^{(k)}(50,x)|$ for $x \in [50,100]$ computed by CR Method (Algorithm 1).

5.1 Small values of the arguments

For small values of the arguments, say $a < x < 100$ the RR method is not applicable. So, we only compare the CR method with Jagerman method. Since, in this range, CR method is much more efficient than Jagerman method, it remains to compare the precision of the calculated values.

Tables 1–4 show some of those results. The approximations calculated are presented together with the percentage of (relative) error assuming that the Jagerman algorithm is exact. In fact, it may be noticed that the Jagerman algorithm is very precise in this range of the arguments. Indeed, comparisons with methods based on adaptive quadrature show results of the Jagerman method in that range with more than 10

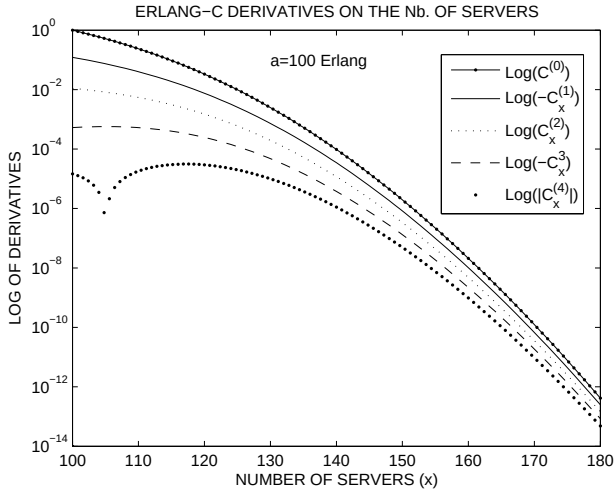

Figure 4: Graphics of $|C_x^{(k)}(100,x)|$ for $x \in [100,180]$ computed by CR Method (Algorithm 1).

exact figures. On the other hand, the precision of the RC method proposed is less than the obtained with the Jagerman method. This fact is easily explained: Gauss-Laguerre quadrature calculates a crude approximation of the initial values and for small values of a and x the relative error of the calculated values does not decay sufficiently since very few steps of the recursion are used. Moreover and as expected, when the order of the derivative increases the accuracy decreases. Table 4 shows that the degradation of the precision of the calculated values by CR method practically disappear for $x > a > 10$. In that table the digits that are not in accordance with the values calculated by Jagerman method are emphasized in bold. Therefore, for $x > a > 10$ it may be concluded that the CR method is much more efficient and allows high precision calculations.

Table 1: Derivatives $C_x^{(k)}$ for $a = 0.5$ and $x = 1$

k	CR Alg.	Jagerman Alg.	Error
0	$5.000\,000\,E-01$	$5.000\,000\,E-01$	0.00000%
1	$-7.307\,315\,E-01$	$-7.307\,277\,E-01$	0.00052%
2	$9.176\,161\,E-01$	$9.175\,930\,E-01$	0.0025%
3	$-8.991\,244\,E-01$	$-8.990\,934\,E-01$	0.0034%
4	$4.946\,733\,E-01$	$4.953\,288\,E-01$	0.13%
5	$2.680\,977\,E-01$	$2.603\,695\,E-01$	3.0%
6	$-8.166\,878\,E-01$	$-7.625\,793\,E-01$	7.1%
7	$1.900\,512\,E-01$	$-9.536\,994\,E-02$	3.0%
8	$1.014\,154\,E+00$	$-2.199\,867\,E+00$	146%
9	$2.828\,523\,E+00$	$-9.486\,902\,E-01$	398%
10	$-1.780\,151\,E+01$	$-9.792\,540\,E+00$	81%

Table 2: Derivatives $C_x^{(k)}$ for $a = 1$ and $x = 2$

k	CR Alg.	Jagerman Alg.	Error
0	$3.333\,333\,E-1$	$3.333\,333\,E-1$	0.000000%
1	$-3.995\,941\,E-1$	$-3.995\,942\,E-1$	0.000010%
2	$4.116\,842\,E-1$	$4.116\,840\,E-1$	0.000054%
3	$-3.323\,324\,E-1$	$-3.323\,319\,E-1$	0.00014%
4	$1.553\,650\,E-1$	$1.553\,671\,E-1$	0.0013%
5	$5.263\,250\,E-2$	$5.260\,938\,E-2$	0.044%
6	$-1.445\,280\,E-1$	$-1.444\,383\,E-1$	0.062%
7	$-4.022\,122\,E-3$	$-4.013\,443\,E-3$	0.22%
8	$2.599\,829\,E-1$	$2.572\,807\,E-1$	1.1%
9	$-1.249\,069\,E-1$	$-1.029\,022\,E-1$	21%
10	$-6.055\,802\,E-1$	$-7.186\,006\,E-1$	16%

Table 3: Derivatives $C_x^{(k)}$ for $a = 2$ and $x = 3$

k	CR Alg.	Jagerman Alg.	Error
0	$4.444\,444\,E-01$	$4.444\,444\,E-01$	0.000000%
1	$-3.882\,138\,E-01$	$-3.882\,138\,E-01$	0.000000%
2	$2.822\,712\,E-01$	$2.822\,712\,E-01$	0.000000%
3	$-1.502\,856\,E-02$	$-1.502\,856\,E-01$	0.000000%
4	$3.366\,232\,E-02$	$3.366\,234\,E-02$	0.000021%
5	$2.656\,584\,E-02$	$2.656\,580\,E-02$	0.00011%
6	$-1.917\,458\,E-02$	$-1.917\,458\,E-02$	0.000024%
7	$-1.800\,746\,E-02$	$-1.800\,673\,E-02$	0.0041%
8	$2.159\,498\,E-02$	$2.159\,062\,E-02$	0.020%
9	$2.458\,913\,E-02$	$2.460\,073\,E-02$	0.047%
10	$-3.935\,980\,E-02$	$-3.934\,333\,E-02$	0.042%

Table 4: Derivatives $C_x^{(k)}$ for $a = 10$ and $x = 12$

k	CR Alg.	Jagerman Alg.
0	$4.493\,882\,242\,982\,E-1$	$4.493\,882\,242\,982\,E-1$
1	$-1.957\,138\,108\,265\,E-1$	$-1.957\,138\,108\,265\,E-1$
2	$6.853\,579\,227\,757\,E-2$	$6.853\,579\,227\,757\,E-2$
3	$-1.630\,475\,310\,872\,E-2$	$-1.630\,475\,310\,872\,E-2$
4	$9.195\,949\,758\,9\mathbf{46}\,E-4$	$9.195\,949\,758\,954\,E-4$
5	$8.915\,293\,473\,\mathbf{794}\,E-4$	$8.915\,293\,473\,783\,E-4$
6	$-7.780\,633\,149\,\mathbf{666}\,E-5$	$-7.780\,633\,149\,575\,E-5$
7	$-1.801\,545\,889\,6\mathbf{23}\,E-4$	$-1.801\,545\,889\,630\,E-4$
8	$1.197\,684\,653\,\mathbf{079}\,E-6$	$1.197\,684\,653\,852\,E-6$
9	$6.757\,700\,191\,\mathbf{225}\,E-5$	$6.757\,700\,191\,184\,E-5$
10	$1.163\,815\,001\,\mathbf{485}\,E-5$	$1.163\,815\,001\,145\,E-5$

5.2 High values of the arguments

For $x > a > 100$ the RR method may be used in order to improve the efficiency of the CR method. The precision of the three method is very good, so the critical criterion of comparison is the efficiency. In order to accomplish that comparison, Figures 5–7 are presented showing log-log graphics of processing times (in milliseconds) versus the magnitude of the argument a. The value of x was pre-calculated such that $W = C(a, x)/(x - a) = 0.5$, meaning that we have a typical Erlang-C system with the mean waiting time is half the mean values of service time. For obtaining better precision for the values of processing times, averages over the values in 1000 runs (RC and Jagerman method) and in 5000 runs (RR method), were computed. In the next paragraph, some conclusions may be drawn from the computational results.

The CR Algorithm is consistently more efficient than the Jagerman method excepting for sufficiently high values of a where the Jagerman approach is more efficient. The critical value of a beyond which that happens seems to be in the range 1000–5000. However, the RR algorithm is much more efficient than the CR algorithm and the obtained approximations are exactly the same. For calculating derivatives until order four, the most efficient method is the RR algorithm providing that $a \leq 3 \times 10^4$.

Nevertheless, in the range $x > a > 10^5$ some methods based on asymptotic approximations may be considered even for high precision computations. Those methods are well known for $C(a, x)$ and for the first derivative $C'_x(a, x)$ (see [4, 15]).

6 Conclusion

Extensive computation has shown that the proposed CR and RR methods are very accurate in a wide range of values of a and x and compares favorably, in terms of efficiency, with the method based in cardinal series quadrature, excepting for very high values of x. For very high values of the arguments (say $x > a > 5\times10^4$) Jagerman algorithm shows better efficiency. Nevertheless, it is important to notice that in this case the precision of the Jagerman algorithm falls down. However in that range of arguments of very high magnitude there are few applications. In these cases, we suggest asymptotic expansion methods.

Considering those features as well as the numerical robustness of the methods we believe that the CR and RR methods proposed in this paper may be attractive and reliable for all ranges of applications (namely in Teletraffic engineering and call centers industry), even if the number of servers is a very large value.

Finally, it is conjectured that $C'_x(a, x)$ is a convex function of x. This numerical evidence is supported by extensive computation covering a wide range of the parameters by showing in all cases that $C'''_x(a, x) < 0$. The theoretical analysis of that conjecture will be carried out elsewhere and may lead to an important property of the Erlang C function.

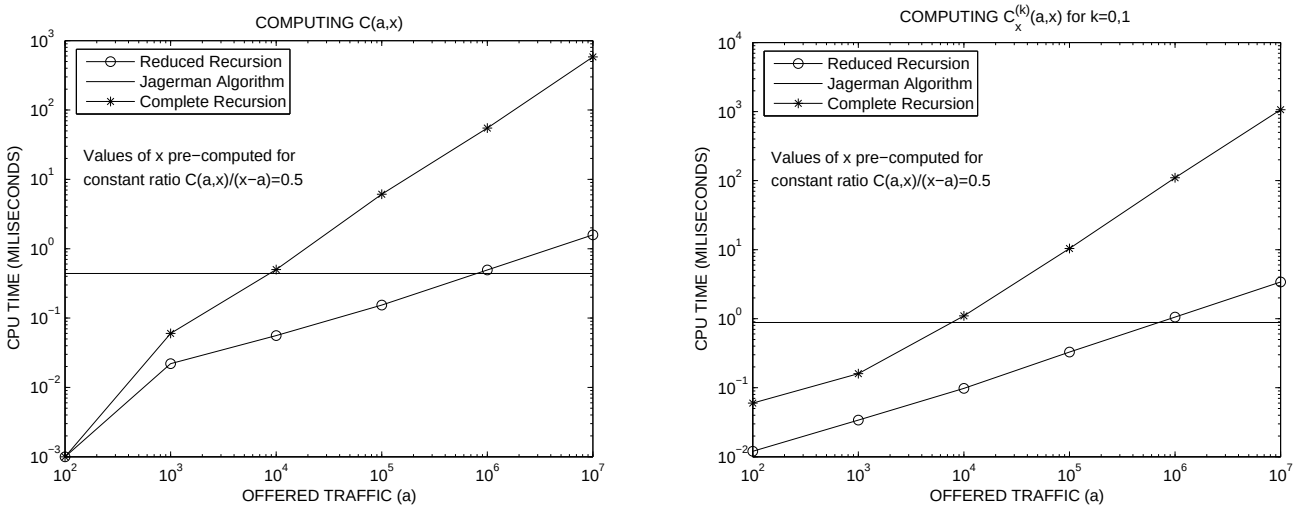

Figure 5: Processing times for the three algorithms.

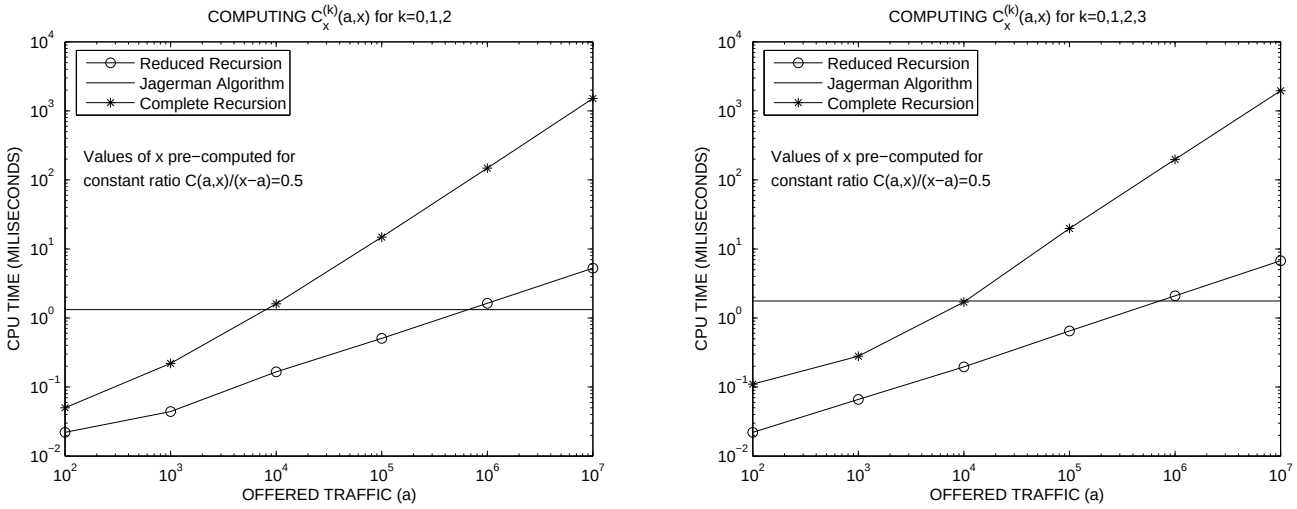

Figure 6: Processing times for the three algorithms.

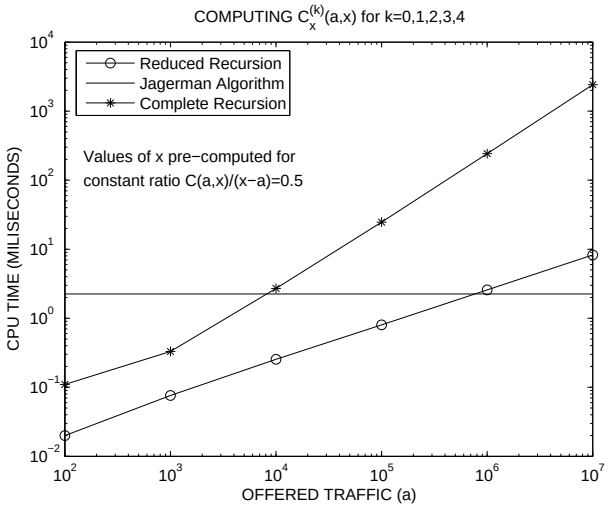

Figure 7: Processing times for the three algorithms.

A Reciprocal Function Derivatives

Here we develop an algorithm to compute the successive derivatives of a *reciprocal function* given by the next Proposition.

Proposition 4 *Let $H : \mathcal{D}_H \subset \mathbb{R} \longrightarrow \mathbb{R}$ be a $n+1$ continuously differentiable function in a neighborhood of a point $y^* \in \text{int}(\mathcal{D}_H)$ and $H(y^*) \neq 0$. The successive derivatives of the function $G(y) = 1/H(y)$ calculated for $y = y^*$ are given for $n = 0, 1, 2, \ldots$ by*

$$G^{(n+1)} = -\sum_{k=0}^{n} \left[\binom{n}{k} \sum_{i=0}^{k} \binom{k}{i} G^{(i)} G^{(k-i)} \right] H^{(n+1-k)}.$$

Proof: The first step in order to get an expression for $G^{(n)}(y^*)$ requires the definition of an appropriate notation able of avoiding unwanted formal complications. Thence, G and H_k may replace $G(y^*)$ and $H^{(k)}(y^*)$ respectively, and:

$$G_k = G^{(k)}(y^*), \quad k = 0, 1, 2, \ldots$$

By differentiation of G it is immediate that $G_1 = -G^2\, H_1$, or, in a more simple way

$$G_1 = \beta_0\, H_1\,, \tag{21}$$

by introducing the notation $\beta = \beta_0 = -G^2$, and:

$$\beta_k = \frac{\partial^k \beta}{\partial\, y^k}, \quad k = 1, 2, \ldots$$

To obtain a general expression for G_n, equation (21) will be successively derived, which requires the calculation of the β_k. These may be expressed as a polynomial in $G_0, \ldots, G_k$, by using the Leibniz formula. Thence, for $k = 0, 1, 2, \ldots$:

$$\beta_k = -\left[\, G \cdot G \,\right]^{(k)} \;=\; -\sum_{i=0}^{k} \binom{k}{i} G_i\, G_{k-i}. \tag{22}$$

Then, from (21) $G_{n+1} = [\beta_0\, H_1]^{(n)}$, or

$$G_{n+1} = \sum_{k=0}^{n} \binom{n}{k} \beta_k\; H_{n+1-k}, \quad n = 0, 1, 2, \ldots \tag{23}$$

The result follows by introducing (22) in (23). □

Algorithm 2 shows the details of the computational scheme in order to compute the expression of Proposition 4. The binomial coefficients present in (22) and (23) may be efficiently computed by recursion.

Algorithm 2 — Computes the expression of Prop. 4.

Require: $H_k = H^{(k)}(y^*)$ for $k = 0(1)m$;
Ensure: $G_k = G^{(k)}(y^*)$ for $k = 0(1)m$;
  $G_0 \leftarrow 1/H_0$;
  for $n = 0$ to $m-1$ **do**
    $\beta_n \leftarrow 0$;
    $H_{n+1} \leftarrow 0$;
    $Q \leftarrow 1$;
    for $k = 0$ to n **do**
      $\beta_n \leftarrow \beta_n - Q \cdot G_k \cdot G_{n-k}$;
      $G_{n+1} \leftarrow G_{n+1} + Q \cdot \beta_k \cdot H_{n+1-k}$;
      $Q \leftarrow (n-k) \cdot Q/(k+1)$;{Binomial coef.;}
    end for
  end for

B Cardinal series Algorithm

Following [16, pp. 3–5, 17–19], a brief review of quadrature theory based on cardinal series will be given. Next the application of the method to the direct evaluation of the integrals $J_k(a, x)$, will be explained. A possible improvement in the method implementation will be considered.

Suppose that $F(u)$ is the Fourier Transform of the function $f \in L^2(\mathbb{R})$:

$$F(u) \;=\; \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)\, \exp(iut)\, dt\,.$$

If:

$$F(u) \;\equiv\; 0 \quad \text{for } u \notin [-\sigma\,,\, +\sigma]\,,$$

the function $f(t)$ is said bandlimited with radian bandwidth σ, and this is denoted by $f \in W_\sigma$.

The formula (see [16]):

$$\int_{-\infty}^{+\infty} f(t)\, dt \simeq h \sum_{j=-\infty}^{+\infty} f(jh)\,, \tag{24}$$

is especially simple since it uses only values of $f(t)$ at the nodal points jh ($j \in \mathbb{Z}$) with equal weights h. From the sampling theorem, the quadrature is exact for $f \in W_\sigma$ provided $h \leq \pi/\sigma$. However, for $f \in L^2(\mathbb{R}) \backslash W_\sigma$, the quadrature rule (24) is in error.

D. L. Jagerman proposes to evaluate non elementary Laplace transforms by means of the method inspired by formula (24). The functions $J_k(a, x)$ may be defined as:

$$J_k(a,x) = \int_0^{+\infty} e^{-t} (1 + t/a)^{x-1} (t/a) [\ln(1 + t/a)]^k\, dt. \tag{25}$$

By recurring to the change of variable $z = \ln t$ one obtains from (25):

$$J_k(a,x) = \int_{-\infty}^{+\infty} \psi_k(z)\, dz,$$

where:

$$\psi_k(z) = \frac{1}{a} \exp\left(2z - e^z + (x-1)\ln(1+e^z/a)\right) \times \times \ln^k(1+e^z/a). \quad (26)$$

Function $\psi(z)$ must be evaluated by expression (26) in order to avoid overflow problems for very high values of the parameters a and x.

The quadrature rule obtained from the cardinal series is based on the following formula:

$$J_k(a,x) \approx h \sum_{j=-\infty}^{+\infty} \psi_k(jh)$$

To define the algorithm, h must be specified and a truncation error associated with the summation process occurs:

$$J_k(a,x) \approx h \sum_{\substack{j\in\mathbb{Z} \\ p_k(jh)>\epsilon}} \psi_k(jh)\,.$$

After some experimentation it is proposed $h = 0.01$ and $\epsilon = 10^{-15}$ for a good trade-off between precision and efficiency of the quadrature algorithm.

Acknowledgements: The research was supported by the *Center for Research on Optimization and Control (CEOC),* University of Aveiro, Portugal, from the "Fundação para a Ciência e a Tecnologia" (FCT), co-financed by the European Community Fund FEDER/POCI 2010.

References:

[1] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions*. Dover Publications, 9th edition, 1970.

[2] H. Akimaru and T. Nishimura. The derivatives of the Erlang's B formula. *Rev. Electr. Commun. Labr., NTT*, 11(9–10):428–445, 1963.

[3] H. Akimaru and T. Nishimura. The derivatives of the Erlang's C formula. *Rev. Electr. Commun. Labr., NTT*, 12(5–6):325–401, 1964.

[4] H. Akimaru and H. Takahashi. Asymptotic expansion for Erlang loss function and its derivative. *IEEE Transactions on Communications*, 29(9):1257–1260, 1981.

[5] L. C. Andrews. *Special Functions of Mathematics for Engineers*. Oxford University Press, 1998.

[6] E. Brockmeyer, H. L. Halstrom, and A. Jensen. *The Life and Works of A. K. Erlang*. Danish Academy of Technical Sciences, Copenhagen, 1948.

[7] S. Chung and J. You. Call admission control in CDMA cellular networks with grade of service and quality of service dimensioning. *WSEAS Transactions on Communications*, 5(12):2182–2189, 2006.

[8] R. Cooper. *Introduction to Queueing Theory*. North Holland, 1981.

[9] A. Costescu, S. Spanulescu, and C. Stoica. Analytical properties and numerical calculations of high transcendental functions involved in the relativistic amplitudes of two photon atomic processes. *WSEAS Transactions on Mathematics*, 8(1):21–31, 2009.

[10] Jorge Sá Esteves, J. Craveirinha, and D. Cardoso. Computing Erlang-B function derivatives in the number of servers — a generalized recursion. *ORSA Communications in Statistics, Stochastic Models*, 11(2):311–331, 1995.

[11] Jorge Sá Esteves, J. Craveirinha, and D. Cardoso. A reduced recursion for computing Erlang-B function derivatives. In *Proceedings of the 15th Int. Teletraffic Congress*, pages 1315–1326. Elsevier Science B. V., 1997.

[12] N. Garnett, A. Mandelbaum, and M. Reiman. Designing a call center with impatient customers. *Manufacturing and Service Operations Management*, 4(3):208–227, 2002.

[13] A. Girard. *Routing and Dimensioning in Circuit-Switched Networks*. Addison-Wesley, 1990.

[14] N. Goans, G. Koole, and A. Mandelbaum. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing and Service Operations Management*, 5(2):79–141, 2003.

[15] D. L. Jagerman. Some properties of the Erlang loss function. *The Bell System Technical Journal*, 53(3):525–551, 1974.

[16] D. L. Jagerman. Mathcalc. Technical Report WPN 311521–0101, FC 40416, AT&T Bell Laboratories Technical Memorandum, March 1987.

[17] D. L. Jagerman and B. Melamed. Models and approximations for call center design. *Methodology and Computing in Applied Probability*, 5:159–181, 2003.

[18] A. A. Jagers and E. A. Van Dorn. Convexity of functions wich are generalizations of the Erlang loss function and the Erlang delay function. *SIAM Review*, 33(2):281–283, June 1991.

[19] G. Koole and A. Mandelbaum. Queueing models of call centers: An introduction. *Annals of Operations Research*, 113:41–59, 2002.

[20] E. Lee, K. Koh, H. Choi, and H. Bahn. On the parallelism of I/O scheduling algorithms in MEMS-based large storage systems. *WSEAS Transactions on Information Science ans Applications*, 6(5):920–923, 2009.

[21] P. M. Papazoglou, D. A. Karras, and R. C. Papademetriou. Improved integral channel allocation algorithms in cellular communication systems enabling multimedia QoS services. *WSEAS Transactions on Communications*, 7(10):1014–1024, 2008.

[22] E. Szybicki. Some numerical methods used for telephone traffic applications. *Ericsson Technics*, 22:203–229, 1964.