delim \$\$

NAME

Berl_i, Berl_d, Xerl, Aerl - Erlang and inverse Erlang functions

SYNOPSIS

#include <erlang.h>

double Berl i(int n, double a);

double Berl_d(double x, double a);

double Xerl(double b, double a);

double Aerl(double x, double b);

DESCRIPTION

These functions give solutions to the Erlang blocking formula. Erlang formula gives the probability that a communications system with n (or x) lines can't handle any more calls. This also depends on the traffic intensity a. All arguments to these functions must be positive and 0 < b < 1.

Berl i() returns the Erlang loss probability for a integer valued trunk group n and Poisson traffic a

Berl_d() returns the Erlang loss probability for a trunk group x, where x is a real valued variable (double). Second parameter is the Poisson traffic a.

Xerl() returns the number of trunks for a loss probability b and Poisson traffic a.

Aerl() returns the Poisson traffic value for a trunk group x and loss probability b.

ALGORITHM

Berl i() is implemented by using the recursion formula

(1) $E(x+1,a) = \{a E(x,a)\} \text{ over } \{x + 1 + a E(x,a)\}.$

Berl_d() is divided into five sections depending on the value of the parameters.

Range 1: x < a and a > 0.01

Apply the continued fraction expansion formula.

Range 2: $x \ge a$ and $x \ge 60$

Determine a value x' such that a > x' (i.e. apply an integral valued shift of x). Call Berl_d again with x' < a and use the recurrence relation to arrive at the original value of x.

Range 3: $x \ge a$ and $1 \le x \le 60$

Apply Rapp's parabola followed by the recurrence relation (1).

Range 4: $x \ge a$ and $a \ge 0.01$ and x < 1

Apply Laguerre quadrature, Abramowitz [2], p. 923

Range 5: 0 < x < 1 and 0 < a <= 0.01

Apply formula (48) from Farmer and Kaufman [ref 1], p. 161

In Xerl() we first determine with inequality

E(n,a) > E(n+1,a)

the position of the interval [N,N+1] such that $N \le X \le N+1$. By using the secant iteration method, we determine the exact root of the function

f(x) = Berl d(x,a) - b.

Evaluation time of **Xerl()** is proportional to the result value.

Aerl(): First we compute a starting value for a. Then a is improved using the Newton's iteration scheme. The iteration process is stopped as soon as the relative error in E(x,a) is smaller than some prescribed value. When b is very small, this function has long evaluation time.

REFERENCES

[1] Farmer, R.F, Kaufman, I., "On the Numerical Evaluation of Some Basic Traffic Formulae", Networks, Vol. 8 (1978) p. 153 - 186

[2] Abramowitz, M., Stegun, I., "Handbook of mathematical functions", Dover Publications, Inc., New York, 8th printing, 1972.