

4400 University Drive
Fairfax, Virginia 22030

4

DTIC FILE COPY

AD-A197 182

George Mason University

COMPUTING EXACT DISTRIBUTION
OF CUSTOMERS IN M^x/D/c QUEUES

BY

M.L. Chaudhry
J.E. Powell
C.M. Harris

DTIC
ELECTE
JUN 27 1988
S D
E

This document has been approved
for public release and sales in
distribution is unlimited.

88 6 27 03 3

4

COMPUTING EXACT DISTRIBUTION
OF CUSTOMERS IN M^x/D/c QUEUES

BY

M.L. Chaudhry
J.E. Powell
C.M. Harris



Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

DTIC
ELECTE
JUN 27 1988
S E D

This document has been approved
for public release and sale; its
distribution is unlimited.

TECHNICAL REPORT

Office of Naval Research

Contract No. N0014-86-K0029

COMPUTING EXACT DISTRIBUTION
OF CUSTOMERS IN M^x/D/c QUEUES

by

M.L. Chaudhry
J.E. Powell
C.M. Harris

Report No. GMU/22474/102
June 21, 1988

Department of Operations Research and Applied Statistics
School of Information Technology and Engineering
George Mason University
Fairfax, VA 22030

Copy No. 12-----

This document has been approved for public sale and release;
its distribution is unlimited.

ADA197 182

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Computing Exact Distribution of Customers in MX/D/c Queues		5. TYPE OF REPORT & PERIOD COVERED Technical Report
7. AUTHOR(s) M.L. Chaudhry, J.E. Powell and C.M. Harris		6. PERFORMING ORG. REPORT NUMBER GMU/22474/102
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research and Applied Statistics George Mason University, Fairfax, Va. 22030		8. CONTRACT OR GRANT NUMBER(s) N0014-86-K0029
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research 800 North Quincy Street Arlington, Va. 22217		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Task H-B Project 4118150
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE June 22, 1988
		13. NUMBER OF PAGES 22
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
applied probability, numerical methods, bulk queueing, rootfinding computational probability, computational analysis probability, probability distributions. stochastic modeling.		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		
Bulk arrival queueing problems arise frequently in the modeling of vehicle loading and telecommunication networks. However, the solution of these problems has been historically hindered by the lack of efficient solution procedures. In this paper, we have discussed an effective numerical approach to solution, and show that it works well for all typical values of the model parameters, C and RHO .		

Computing Exact Distribution of Customers in $M^X/D/c$ Queues

by

M.L. Chaudhry, J.E. Powell

Department of Mathematics and Computer Science

Royal Military College of Canada, Kingston, Ontario, K7K 5L0, Canada

and C. Harris

Department of Operations Research and Applied Statistics

George Mason University, Fairfax, VA 22030, U.S.A.

Keywords: Bulk Queues, Roots

Language

Fortran 77

Description

Just about the most common multi-server, batch-arrival queueing system encountered in practice is the bulk Poisson input and constant service model, symbolized by $M^X/D/c$. A prototypical application arises in vehicle loading problems: batches arrive as a Poisson process to be loaded onto numerous vehicles at a fixed rate. The analytic development of the model for the bulk-arrival, multiserver queue $M^X/D/c$ is given in Chaudhry and Templeton (1983). Further computational aspects of the computation of the distribution of customers have been discussed in Chaudhry et al. (1988). Smith (1987) considered only an approximate solution to a special case of $M^X/D/c$, viz., the non-bulk $M/D/c$ queue. We present here an exact numerical solution for the distribution of the number in the system for the queueing model $M^X/D/c$.

Many applications of the model $M^X/D/c$ may involve high values of the model parameters, particularly if the model is applied to large-scale systems in telecommunication and transportation. The purpose of this paper, therefore, is to complement the work reported by Chaudhry et al. (1988) by producing a refined version of the program used for that paper. The program will run for high values of the bulk arrival parameters (group sizes ≤ 100), high and low values of c and ρ ($1 \leq c \leq 100$, $.01 \leq \rho \leq .9$). Also, limited testing has indicated that the program should work for more extreme values as well.

The probability generating function (p.g.f.) for the bulk-arrival, multi-server system $M^X/D/c$ is inverted numerically by finding the roots of the characteristic equation (c.e.) of the p.g.f.. The roots are found here using a special algorithm that combines the bisection and Muller's methods. The probabilities for the number in the system are calculated by using recursive formulas given by equations (2),(3),(6),(7), and (8). Then if needed (for example, when c is "large" or there is a "large" number of expected arrivals during service), the formulas given by (4), and (5) can be used in an iterative way to increase the accuracy of the probabilities.

Formulas:

Using the notation found for example in Chaudhry and Templeton (1983):

λ = arrival rate; b = duration of service time (constant); c = number of servers;

a_j = Prob(group size, $X = j$); $\bar{a}$ = mean batch size = $\sum j a_j$; ρ = utilisation factor = $\lambda \bar{a} b / c$;

$\pi_n(t)$ = Prob[total number of arrivals during $(0, t) = n$]; P_n = Prob[n customers are in the system]

The underlying c.e. of the model is:

$$z^c - \exp\{b\lambda(A(z) - 1)\} = 0 \quad (1)$$

which has exactly $(c - 1)$ distinct roots within and one root on the unit circle, $|z| = 1$. Let the roots be $z_1, \dots, z_{c-1}, z_c = 1$. Chaudhry et al. (1988) show that the first c steady-state system size probabilities in $M^X/D/c$ are given by

$$P_0 = c(1 - \rho) \prod_{i=1}^{c-1} \frac{z_i}{(z_i - 1)}, \quad (2)$$

$$P_n = b_n P_0, \quad 1 \leq n < c \quad (3)$$

with b_n = coefficient of z^n in $\prod_{i=1}^c (1 - z/z_i)$. Then, using the closed-form equations for π_n (also given in Chaudhry and Templeton, 1983) and the steady-state Chapman-Kolmogorov (C-K) equations written as

$$P_0 = \pi_0(b) \sum_{m=0}^c P_m, \quad (4)$$

$$P_n = \pi_n(b) \sum_{m=0}^c P_m + \sum_{m=c+1}^{c+n} P_m \pi_{n-m+c}(b), \quad n = 1, 2, \dots \quad (5)$$

we find that

$$P_c = P_0 \exp(c\rho/a) - \sum_{m=0}^{c-1} P_m, \quad (6)$$

$$P_{c+1} = \frac{P_1 - \pi_1(b) \sum_{m=0}^c P_m}{\pi_0(b)} \quad (7)$$

and

$$P_{c+n} = \frac{P_n - \pi_n(b) \sum_{m=0}^c P_m - \sum_{m=c+1}^{c+n-1} P_m \pi_{n-m+c}(b)}{\pi_0(b)}, \quad n = 2, 3, \dots \quad (8)$$

Method for Calculating P_n

The probabilities are calculated by first finding the roots of (1) and then using the recursive relations given by equations (2),(3),(6),(7), and (8). The calculations proceed until either the sum of the probabilities in the vector $\{P_n\}$ is very close to one or the maximum dimension of the probability vector $\{P_n\}$ is reached.

The C-K equations (4) and (5) are applied to the probability vector as an iteration in order to increase the accuracy of the probabilities whenever the recursive relations fail to achieve satisfactory results. The probability vector is then iterated until either one of two conditions is met. If we let $\{PQB_n\}$ = probability vector before an iteration, and $\{P_n\}$ = probability vector after an iteration, then the basic condition is:

$$DIFF = \sum_{n=0}^{size} |P_n - PQB_n| < DIFM$$

where *size* is the size of the probability vector $\{P_n\}$, and *DIFM* is a tolerance to be given by the user as an input parameter. The other condition is that the number of iterations cannot exceed the input parameter *MAXIT*.

In most cases, the recursive method will produce very accurate results and the iterative method will not be needed. However, the iterative method can still be used as a good check of the accuracy of the probabilities. If the probability vector is iterated on and *DIFF* is very small (say $< 10^{-9}$), then the iterative method has converged and you have very accurate results. When P_0 is small, the recursive formula will not yield very accurate results but will still produce a good starting vector for the iterative formula. When $P_0 < 10^{-30}$, the approximation $P_0 = 10^{-30}$ is used in order to get some rough starting vector.

The system $M^X/D/c$ has an interesting property that has been used to speed up the calculations. If

N is the greatest common divisor between the different group sizes and the number of servers, then only every N^{th} probability will be non-zero. Thus the program has to only calculate every N^{th} probability.

Method for Finding the Roots

Since the basic root-finding method is discussed in Chaudhry et al. (1988), it is briefly given here.

The roots of (1) that are lying on the real axis inside the unit circle are found by using a combination of the bisection and Muller's method for finding roots. Then the complex roots in the upper half plain of the unit circle are found by using a special algorithm. In order to use this algorithm, equation (1) has to be modified. Multiplying the R.H.S. of (1) by $e^{2\pi in}$, writing $z = re^{i\theta}$ and taking logarithms, we can rewrite (1) as the two equations

$$c \ln r = R \left(\frac{c\rho}{\alpha} (A(z) - 1) \right) \quad (9)$$

and

$$c\theta + 2\pi n = I \left(\frac{c\rho}{\alpha} (A(z) - 1) \right) \quad (10)$$

where $R(w)$ and $I(w)$ denote the real and imaginary parts of w respectively. The algorithm is then for $j = 1, 2, \dots, \frac{c-m}{2}$ (where m is the number of real roots):

- (i) fix θ at $\theta_\alpha = \theta_{j-1} + \delta$ where $\theta_0 = 0$ and δ is an increment initially defined as $\pi/(c+1)$ and later (for $j \geq 2$) adjusted to $\min((\theta_{j-1} - \theta_{j-2})/2.0, \pi/(c+1))$.
- (ii) solve (9) for r_{θ_α} by using a combination of the bisection and Muller's methods.
- (iii) evaluate the corresponding value of n_{θ_α} , the solution of (10) for n using θ_α and r_{θ_α} .
- (iv) fix θ at $\theta_\beta = \theta_{j-1} + 2\delta$, and obtain r_{θ_β} and n_{θ_β} .

Defining $f(t)$ as the fractional part of t , the algorithm now repeats with $\theta_\alpha = \theta_\beta$ and $\theta_\beta = \theta_\beta + \delta$ if $f(n_{\theta_\alpha}) \leq f(n_{\theta_\beta})$; otherwise there is a root between θ_α and θ_β . Knowing there is a root between θ_α and θ_β , we then obtain r_{θ_γ} and n_{θ_γ} for $\theta_\gamma = .5(\theta_\alpha + \theta_\beta)$, and if $f(n_{\theta_\gamma}) > f(n_{\theta_\alpha})$, set $n_{\theta_\alpha} = n_{\theta_\gamma}$ and $\theta_\alpha = \theta_\gamma$; otherwise $n_{\theta_\beta} = n_{\theta_\gamma}$ and $\theta_\beta = \theta_\gamma$. This process of bisecting the sector $(\theta_\alpha, \theta_\beta)$ continues until $\theta_\beta - \theta_\alpha < 10^{-4}$. Then the value $z_j = r_{\theta_\gamma} \exp(i\theta_\gamma)$ is a root of (9) and (10). However, because of the natural logarithmic function, this approximation needs to be refined a bit more. Therefore, this root is used as starting value for Muller's method using the c.e. which will yield a more accurate root.

If the angle being looked at is greater than π before all the roots in the upper half plane are found, then the algorithm has missed a root. In these rare cases, the algorithm is repeated using a smaller δ and a smaller increment for the bisection method. This increases the computation time but does overcome the problem.

Structure

SUBROUTINE MXDC(NNZ,NA,PA,ABAR,C,RHO,MAXIT,DIFM,RTS,P,SIZE)

Input Variables:

NNZ	Integer	Number of arriving groups
NA	Integer array(GS)	Group Sizes (indexed from $i=1, \dots, \text{NNZ}$ in order of size) (eg. $\text{NA}(1) = 10, \text{NA}(2) = 30, \text{NNZ}=2$)
PA	Real array(GS)	Prob(group size of $\text{NA}(i)$ arrives) (eg. $\text{PA}(1) = 0.4, \text{PA}(2) = 0.6$)
ABAR	Real	Mean group size = $\bar{a}$
C	Integer	Number of Servers = c
RHO	Real	Utilization factor = ρ
MAXIT	Integer	Maximum number of iterations
DIFM	Real	Maximum difference between consecutive probability vectors

Output Variables:

MAXIT	Integer	Number of iterations performed
DIFM	Real	Difference between corresponding probability vectors If $\text{DIFF} = -1$, then the recursive method was used.
RTS	Complex array(RT)	Roots of the c.e.
P	Real array(PS)	Prob(number in the system = i)
SIZE	Integer	Size of the probability vector.

The sizes of the arrays are specified in PARAMETER statements in the subroutines and can easily be changed. The maximum sizes of the root vector, probability vector and number of group sizes are in RS,

PS, and GS respectively. The program has not been thoroughly tested for cases where RS is larger than 100. GS can be increased and should not cause problems. The accuracy of the iterative method is dependent on the size of PS (see the accuracy section). The value of 3000 for PS has proved to be large enough in most cases tried.

This subroutine calls upon the following subroutines; GETRAD, MULLER, MULERS, and FN. GETRAD is used by the root-finding section of the program. Given a particular angle and function, this routine will find the radius that will give a zero of the function. MULLER is a version of Muller's method that finds only real roots. It is used by GETRAD. MULERS is a version of Muller's method that finds complex roots and is used by the second part of the root finding section of the program. FN computes values of c.e. and real and imaginary parts of the ln of the c.e.. This is used by the root finding section of the program and is discussed in Chaudhry et al. (1988).

Accuracy and Limitations

The accuracy of the root-finding algorithm is measured by comparing the value of the c.e. at the roots to the value of z^c at the roots. Given a root z , if $|z^c - \exp\{b\lambda(A(z) - 1)\}| \ll |z^c|$, then z is a good root. The accuracy of the probabilities is determined by comparing means (L) and variances (Var) as calculated by using: the roots, the first c probabilities, and all the probabilities. These formulas are discussed by Chaudhry et al. (1988) and were used to produce the enclosed table 1. Another measure of accuracy is the value of the output variable DIFM. This output variable is equal to the sum of the absolute values of the differences between corresponding probabilities in consecutive probability vectors. Thus, the smaller the value of DIFM, the greater the convergence of the iterative method to a steady-state solution. In most cases tried, after two iterations, the means and the variances matched to four or more significant figures, and $DIFM < 10^{-3}$.

Two other conditions that are required for accurate probabilities are that the sum of the probabilities should be very close to one, and the size of the probability vector should not be equal to the maximum vector size PRSIZE. These conditions are required for the iterative formula to be valid. When ρ is very high, the size of the probability vector can sometimes exceed the memory capacity of the computer. In these cases, the prematurely truncated vector will typically converge slowly to some steady-state solution, which may or

may not be close to the actual solution.

Time

The time required to find the roots ranged from only taking a few seconds to taking a few minutes (on an IBM PC AT with 8087 numeric co-processor), depending on how many roots had to be found. The time required to calculate the probabilities varied drastically depending on the queuing parameters and the accuracy required. For reasonable accuracy (i.e., having the means and the variances match to within 4 significant figures and $DIFM < 10^{-3}$) the time required would be in the order of a few minutes. This time would increase if greater accuracy was required, or if very high values for c and ρ were used.

Precision

Single precision should be adequate for machines using at least 64 bits to represent real numbers. Otherwise double precision is required. In order to change the program to double precision, the explicit REAL statement has to be changed to DOUBLE PRECISION, and the functions REAL, EXP, CEXP, ABS, CABS, AIMAG, CONJG, SQRT, ACOS, COS, SIN, LOG, replaced by DREAL, DEXP, CDEXP, DABS, CDABS, DIMAG, DCONJ, DSQRT, DACOS, DCOS, DSIN, DLOG. Also all the constants throughout the program have to be changed to double precision (ie. 1.0 changes to 1.0D0).

Example

A sample run is given in table 1. All the input variables and some of the output variables are shown.

Acknowledgements

This project was partially supported by CRAD research grant number FUHDH and FE 1430-8730-770. We wish to thank Natalie Rissesco, Joe Michel, and Gilles Brière for their contributions to the development of the method described here. Part of the work on this project was done at the Department of Operations Research and Applied Statistics (ORAS), George Mason University, where the first author (M.L. Chaudhry) held a visiting professorship. He is grateful to the Department of ORAS where all facilities conducive to research were provided. The work of Carl Harris on this paper was partially supported by ONR Grant N0014-86-K-0029.

References

- Chaudhry, M.L. and J.G.C. Templeton (1983) *A first course in bulk queues*. New York: Wiley.
- Chaudhry, M.L., J.G.C. Templeton, J. Medhi (1988) Computational Analysis of Multiserver Bulk-Arrival Queues with constant Service time $M^X/D/c$. unpublished.
- Smith, V.L. (1987) Algorithm AS 230 Distribution of Customers in $M/E_T/m$ Queues using Hokstad's Approximation. *J. App. Stat.*, 394-401.

Table 1 Example of Input/Output for $M^X/D/10$, with $\rho = 0.5$, $a_1 = 0.5$,
 $a_2 = 0.25$, $a_3 = 0.125$, $a_4 = 0.125$

Input Variables:

NNZ = 4	MAXIT = 5	ABAR = 1.875
NA(1) = 1	PA(1) = 0.5	C = 10
NA(2) = 2	PA(2) = 0.25	RHO = 0.5
NA(3) = 3	PA(3) = 0.125	DIFM = 1.0E-9
NA(4) = 4	PA(4) = 0.125	

Output Variables:

MAXIT = 5	P(3) = 0.1079159
DIFM = 0.8702723E-06	P(4) = 0.1161159
RTS(1) = (1.0000000, 0.0000000)	P(5) = 0.1068249
RTS(2) = (-0.7178427, 0.0000000)	P(10) = 0.0380577
RTS(3) = (0.5857134, 0.5578024)	P(20) = 0.0008178
RTS(4) = (0.1643529, 0.7369143)	P(30) = 0.8036800E-05
RTS(5) = (-0.2728908, 0.6757375)	P(40) = 0.7397169E-07
RTS(6) = (-0.5893607, 0.4034265)	P(50) = 0.6576123E-09
RTS(7) - RTS(10) are conjugates	P(60) = 0.5322294E-11
of RTS(3) - RTS(6)	P(70) = 0.2077907E-13
P(0) = 0.0630749	P(72) = 0.1221037E-13
P(1) = 0.0860505	SIZE = 72
P(2) = 0.1021267	

Check on Output:

L (roots) = 5.2860534	Var(roots) = 13.6705954
L (first c prob.) = 5.2860540	Var(first c prob.) = 13.6705917
L (all the prob.) = 5.2860342	Var(all the prob.) = 13.6704229

Sum of the Probabilities = 0.9999989

Note: A user-friendly software package for the IBM PC family of computers (or compatibles) has been developed by the authors. For more information about this and other queuing packages please contact A&A publications, 395 Carrie Crescent, Kingston, Ontario, K7M-5X7, Canada.

```

        SUBROUTINE MXDC(NNZ,NA,PA,ABAR,C,RHO,MAXIT,DIFM,RTS,P,
6 SIZE)
C
C*****
C      This program is designed to compute probabilities  *
C      of the number in the system for the bulk-arrival,  *
C      multiserver queue  $M^x/D/c$ . *
C*****
C
      INTEGER PS,RS,GS
      PARAMETER(PS=3000,RS=510,GS=31)
      INTEGER NRR,NCR,I,C,SIZE,NUMIT,MAXIT,IBEG,FLAG
      INTEGER NNZ,J,N,M,KSIZE,NA(GS),IGCD,GCD
      REAL F1,F2,FF,X,RHO,ABAR,ANGSP,ANG,PQB(0:PS),F,P(0:PS)
      REAL PA(GS),RAD1,RAD2,ANGO,DIFF,DIFM,CDF,PI(0:PS),SUM3
      REAL RADIUS,ANG1,ANG2,ANGLE,TOL,PI2,SUM,SUMTOL,RUNSUM
      COMPLEX RTS(RS),COEF(RS),RT,CF,PC
      PARAMETER(TOL=1.E-14)
C
C INITIALIZE VARIABLES AND FIND THE REAL ROOTS
C
      PI2=4.0*ACOS(0.0)
      ANG=2.0*ACOS(0.0)/(C+1)
      X=0.005
5      ANGLE=0.0
      DIFF=-1.0
      NUMIT=0
      NRR=0
      RAD1=-1.0-TOL
      RAD2=1.0-TOL
10     CALL GETRAD(GS,RADIUS,ANGLE,RAD1,RAD2,X,1,FLAG,F,PA,NA,NNZ,C,RHO
6 ,ABAR,RT,CF)
      IF (FLAG.EQ.1) THEN
          NRR=NRR+1
          RTS(NRR)=RADIUS
          RAD1=RADIUS+2*X
          GOTO 10
      ENDIF
      NRR=NRR+1
      IF(NRR.NE.1) RTS(NRR)=RTS(1)
      RTS(1)=(1.0,0.0)
C
C Find Complex Roots
C
C
      X=X*10
      ANGO=0.0
      NCR=0
      RAD1=2*X

```

```

RAD2=1-TOL
X=X*.8
IF(NRR.EQ.C) GOTO 90
ANG1=ANG/2
C      start of loop
20      F1=0.0
        DO 30 ANGLE=ANG1,PI2/2,ANG
            CALL GETRAD(GS,RADIUS,ANGLE,RAD1,RAD2,X,3,FLAG,FF,PA,NA,
6      NNZ,C,RHO,ABAR,RT,CF)
            RAD1=RADIUS*.8
            CALL FN(GS,PA,NA,NNZ,C,RHO,ABAR,F2,ANGLE,RADIUS,4,RT,CF)
            IF(F1.GT.F2) GOTO 40
            F1=F2
30      CONTINUE
        GOTO 60
40      ANG1=ANGLE-ANG
        ANG2=ANGLE
50      IF(ANG2-ANG1.GT.1.0E-4) THEN
            ANGLE=(ANG1+ANG2)/2.0
            CALL GETRAD(GS,RADIUS,ANGLE,RAD1,RAD2,X,3,FLAG,FF,PA,NA,
6      NNZ,C,RHO,ABAR,RT,CF)
            RAD1=RADIUS*.8
            CALL FN(GS,PA,NA,NNZ,C,RHO,ABAR,FF,ANGLE,RADIUS,4,RT,CF)
            IF(FF.LT.F1) THEN
                ANG2=ANGLE
                F2=FF
            ELSE
                ANG1=ANGLE
                F1=FF
            ENDIF
            GOTO 50
        ENDIF
        CALL MULERS(GS,ANGLE,RADIUS,FF,PA,NA,NNZ,C,RHO,ABAR,RT,CF)
        NCR=NCR+1
        RTS(NCR+NRR)=RT
        IF((NCR*2+NRR).LT.C) THEN
            ANGSP=ANGLE-ANGO
            ANGO=ANGLE
            IF (ANGSP.LT.ANG) ANG=ANGSP
            ANG1=ANGLE+ANG/2
            GOTO 20
        END IF
C      end of loop
        GOTO 70
C
C      Missed some roots -- try again with smaller step size
C
60      ANG=ANG/2.0
        X=X/50.0
        GOTO 5
C
C      FINISHED ROOT FINDING - COMPUTE CONJ.
C
70      DO 80 N=1,NCR
            RTS(NCR+NRR+N)=CONJG(RTS(NRR+N))
80      CONTINUE

```

```

C
C                                     CALCULATE THE PROBABILITIES P(n)
C
C      Intialize Variables
C
90      SUMTOL=-1.0 -1.E-12
        SIZE=0
        IBEG=0
        DO 100 J=0,PS
            PQB(J)=0.0
            PI(J)=0.0
            P(J)=0.0
100     CONTINUE
        DO 110 I=1,C
            COEF(I)=0.0
110     CONTINUE
C
C      Find the Greatest Common Divisor
C
        GCD=0
        DO 130 N=NA(1),1,-1
            DO 120 J=1,NNZ
                IF((NA(J)/N)*N.NE.NA(J)) GOTO 130
120     CONTINUE
            IF(GCD.EQ.0) GCD=N
            IF((C/N)*N.EQ.C) GOTO 140
130     CONTINUE
140     IGCD=N
C
C      Calculate PI's
C
        KSIZE=PS
        PI(0)=EXP(-C*RHO/ABAR)
        SUM=PI(0)
C
        DO 170 N=GCD,(PS-1),GCD
            PI(N)=0.0
            DO 150 I=1,NNZ
                IF((N-NA(I)).LT.0) GOTO 160
                PI(N)=PI(N)+NA(I)*PA(I)*PI(N-NA(I))
150     CONTINUE
160     PI(N)=PI(N)*C*RHO/(ABAR*N)
            SUM=SUM+PI(N)
            IF(SUM.GE.SUMTOL) THEN
                SUM=SUM-PI(N)
                KSIZE=N
                GOTO 180
            ENDIF
170     CONTINUE
        KSIZE=N-GCD
180     DO 190 N=KSIZE,PS
            PI(N)=0.0
190     CONTINUE
C
C      CALCULATE P(0)
C

```



```

      PC=1.0
      DO 230 I = 2,C
        PC = PC*RTS(I)/(RTS(I)-1.0)
230    CONTINUE
      P(0) =REAL(PC)*C*(1.0-RHO)
      IF(P(0).LE.1.0E-30) THEN
        P(0)=1.0E-30
      ENDIF
C
C CALCULATE THE COEFFICIENTS FOR THE FIRST C PROBABILITIES
C
      COEF(1) = -1.E0/RTS(1)
      DO 250 I = 2, C
        DO 240 J = 2, I
          COEF(I-J+2) = COEF(I-J+2) - COEF(I-J+1)/RTS(I)
240      CONTINUE
        COEF(1) = COEF(1) - 1.E0/RTS(I)
250    CONTINUE
C
C CALCULATE P(1) ---> P(C-1)
C
      SUM=P(0)
      IF (C.EQ.1) GOTO 270
      DO 260 I = IGCD, C-1,IGCD
        P(I)=REAL(COEF(I))*P(0)
        IF(P(I).LT.0.0.OR.P(I).GT.1.0) THEN
          SIZE=I
          P(I)=0.0
          GOTO 300
        END IF
        SUM=SUM+P(I)
        IF(SUM.GE.SUMTOL)THEN
          SIZE=I
          GOTO 300
        ENDIF
260    CONTINUE
C
C CALCULATE P(C),P(C+1),P(C+IGCD) ---> P(SIZE)
C
270    P(C)=P(0)*EXP(C*RHO/ABAR) - SUM
      IF(P(C).LT.0.0) P(C)=0.0
      SUM=SUM+P(C)
      P(C+1)=(P(1)-PI(1)*SUM)/PI(0)
      IF(IGCD.EQ.1) THEN
        IBEG=2
      ELSE
        IBEG=IGCD
      ENDIF
      CDF=SUM+P(C+1)
      DO 290 N=IBEG,(PS-C),IGCD
        IF(N-IGCD.LT.KSIZE) THEN
          I=C+IGCD
        ELSE
          I=N+C-KSIZE
        ENDIF
        SUM3=0.0

```

```

DO 280 M=I,C+N-1,IGCD
    SUM3=SUM3+P(M)*PI(N-M+C)
280    CONTINUE
    P(C+N)=(P(N)-PI(N)*SUM-SUM3)/PI(0)
    IF(P(C+N).LT.0.0)P(C+N)=0.0
    CDF=CDF+P(C+N)
    IF(CDF.GE.SUMTOL)THEN
        SIZE=C+N
        GOTO 300
    ENDIF
290    CONTINUE
    SIZE=PS
C
C PREPARE FOR ITERATION LOOP
C
300    SUM=0.0
    DO 310 N=0,SIZE
        PQB(N)=P(N)
        SUM=SUM+PQB(N)
310    CONTINUE
    IF(SUM.LT.0.9999) THEN
        DO 320 N=0,SIZE
            PQB(N)=PQB(N)/SUM
            P(N)=PQB(N)
320    CONTINUE
        ENDIF
C
C START OF ITERATION LOOP
C
    DO 390 NUMIT=1, MAXIT
        SUM=0.0
        DO 330 I=0,C-1
            SUM=SUM+PQB(I)
330    CONTINUE
        P(0)=PI(0)*(SUM+PQB(C))
        RUNSUM=P(0)
        DO 360 N=IGCD,PS,IGCD
            SUM3=0.0
            DO 340 J=MIN0(KSIZE,N),0,-1
                IF((N-J+C).GT.(SIZE-1)) GOTO 350
                SUM3=SUM3+PQB(N-J+C)*PI(J)
340    CONTINUE
350    P(N)=PI(N)*SUM+SUM3
            IF((RUNSUM+P(N)).GT.SUMTOL) GOTO 370
            RUNSUM=RUNSUM+P(N)
            IF(P(N).LT.1.E-14.AND.RUNSUM.GT.0.9999) GOTO 370
360    CONTINUE
370    SIZE=N-IGCD+1
        DIFF=0.0
        DO 380 N=0,SIZE-1,IGCD
            DIFF=ABS(P(N)-PQB(N))+DIFF
            PQB(N)=P(N)
380    CONTINUE
            IF(DIFF.LE.DIFM) GOTO 400
390    CONTINUE

```

```

      NUMIT=NUMIT-1
C
C  RETURN TO MAIN PROGRAM
C
400  DIFM=DIFF
      MAXIT=NUMIT
      RETURN
      END
C
C
C  -----
C
      SUBROUTINE GETRAD(GS,RADIUS,ANGLE,RAD1,RAD2,X,N,FLAG,FF,PA,NA,
6    NNZ,C,RHO,ABAR,RT,CF)
C
C  -----
C
      INTEGER GS
      INTEGER FLAG,N,NA(GS),NNZ,C
      REAL ANGLE,RADIUS,RAD1,RAD2,X,FF,F1,RHO,ABAR,PA(GS)
      COMPLEX RT,CF
C
      FLAG=1
      RADIUS=RAD1
      CALL FN(GS,PA,NA,NNZ,C,RHO,ABAR,F1,ANGLE,RADIUS,N,RT,CF)
10    RADIUS=RADIUS+X
      IF(RADIUS.GT.RAD2) GOTO 20
      CALL FN(GS,PA,NA,NNZ,C,RHO,ABAR,FF,ANGLE,RADIUS,N,RT,CF)
      IF(FF*F1.LT.0.0) THEN
        RADIUS=RADIUS-X/2.0
        GOTO 30
      ENDIF
      F1=FF
      GOTO 10
20    FLAG=0
      RADIUS=1.0-1.0E-10
30    CALL MULLER(GS,ANGLE,RADIUS,F1,FF,N,X,PA,NA,NNZ,C,RHO,ABAR,RT,CF)
40    RETURN
      END
C
C
C  -----
C
      SUBROUTINE MULERS(GS,ANGLE,RADIUS,FF,PA,NA,NNZ,C,RHO,ABAR,RT,
6    CFRTS)
C
C  -----
C
      THIS ROUTINE IS THE COMPLEX-VALUED FORM OF MULLERS
      METHOD USED TO FIND ROOTS OF THE CHAR. EQN.
C
      INTEGER GS
      INTEGER NA(GS),NNZ,C
      COMPLEX CI,H,Z,DELFP,CFRTS,FRTPRV,FRTDEF,LAMBDA
      COMPLEX DELF,DFPRM,NUM,G,SQR,DEN,RT
      REAL RADIUS,PI2,ANGLE,TOL1,TOL3,FF,RHO,ABAR,PA(GS)

```

```

PARAMETER(CI=(0.0,1.0),TOL3=1.0E-9,TOL1=1.0E-16)
C
PI2=4*ACOS(0.0)
KCOUNT=-1
KOUNT=0
KCOUNT=KCOUNT+1
IF(KCOUNT.EQ.7) GOTO 40
MAXIT=25
Z=RADIUS*CEXP(CI*ANGLE)
H=.00001E0*10.E0**KCOUNT
RT=Z+H
CALL FN(GS,PA,NA,NNZ,C,RHO,ABAR,FF,ANGLE,RADIUS,2,RT,CFRTS)
DELFPR=CFRTS
RT=Z-H
CALL FN(GS,PA,NA,NNZ,C,RHO,ABAR,FF,ANGLE,RADIUS,2,RT,CFRTS)
FRTPRV=CFRTS
RT=Z
CALL FN(GS,PA,NA,NNZ,C,RHO,ABAR,FF,ANGLE,RADIUS,2,RT,CFRTS)
FRTDEF=CFRTS
DELFPR=FRTPRV-DELFPR
LAMBDA=-0.5E0
10 DELF=FRTDEF-FRTPRV
DFPRLM=DELFPR*LAMBDA
NUM=-FRTDEF*(1.0+LAMBDA)*2.0
G=(1.0+LAMBDA*2.0)*DELF-LAMBDA*DFPRLM
SQR=G*G+2.0*NUM*LAMBDA*(DELF-DFPRLM)
IF((REAL(SQR).LT.0.0).AND.(ANGLE.GT.(PI2/2.E0-0.001E0)))
6 SQR=(0.0,0.0)
SQR=CSQRT(SQR)
DEN=G+SQR
IF(ANGLE.GT.(PI2/2.E0-0.001E0))THEN
  IF((REAL(G)*REAL(SQR)).LT.0.0)DEN=G-SQR
  GOTO 20
ENDIF
IF(REAL(G)*REAL(SQR)+AIMAG(G)*AIMAG(SQR).LT.0.0)DEN=G-SQR
20 IF(CABS(DEN).LT.TOL3)DEN=1.E0
LAMBDA=NUM/DEN
FRTPRV=FRTDEF
DELFPR=DELF
H=H*LAMBDA
RT=RT+H
IF(KOUNT.GT.MAXIT) GOTO 40
KOUNT=KOUNT+1
30 IF(CABS(RT).GT.1.2E0) THEN
  ANGLE=ACOS(REAL(RT)/CABS(RT))
  RT=.99999999*CEXP(CI*ANGLE)
ENDIF
CALL FN(GS,PA,NA,NNZ,C,RHO,ABAR,FF,ANGLE,RADIUS,2,RT,CFRTS)
FRTDEF=CFRTS
IF(CABS(H).LT.TOL1*CABS(RT)) GOTO 40
IF(CABS(CFRTS).LT.TOL1) GOTO 40
IF(CABS(FRTDEF).LT.10.0*CABS(FRTPRV)) GOTO 10
H=H/2.0
LAMBDA=LAMBDA/2.0
RT=RT-H
GOTO 30

```

```

40  RADIUS=CABS(RT)
    ANGLE=ACOS(REAL(RT)/RADIUS)
    RETURN
    END
C
C
C  -----
C
    SUBROUTINE MULLER(GS,ANGLE,RADIUS,F1,FF,N,X,PA,NA,NNZ,C,RHO,ABAR,
6   RT,CF)
C
C  -----
C
    THIS ROUTINE IS THE REAL-VALUED FORM OF
    MULLERS METHOD
C
    INTEGER GS
    REAL FRTPRV,DELFPR,FRTDEF,DELF,DFPRLM,G,SQR,H,DEN
    REAL LAMBDA,NUM,F1,RHO,ABAR,PA(GS),RADIUS,ANGLE,FF,X
    INTEGER KOUNT,MAXIT,N,NA(GS),NNZ,C
    COMPLEX RT,CF
C
    KOUNT=0
    MAXIT=25
    H=X*0.5E0
    FRTPRV=F1
    DELFPR=F1-FF
    CALL FN(GS,PA,NA,NNZ,C,RHO,ABAR,FF,ANGLE,RADIUS,N,RT,CF)
    FRTDEF=FF
    LAMBDA=-0.5E0
10  DELF=FRTDEF-FRTPRV
    DFPRLM=DELFPR*LAMBDA
    NUM=-FRTDEF*(1.0+LAMBDA)*2.0
    G=(1.0+LAMBDA*2.0)*DELF-LAMBDA*DFPRLM
    SQR=G*G+2.0*NUM*LAMBDA*(DELF-DFPRLM)
    IF(SQR.LT.0.0)THEN
        LAMBDA=G/(2.0*LAMBDA*(DFPRLM-DELF))
    ELSE
        SQR=SQRT(SQR)
        DEN=G+SQR
        IF(G*SQR.LT.0.0)DEN=G-SQR
        IF(ABS(DEN).LT.1.E-7) DEN=1.0
        LAMBDA=NUM/DEN
    ENDIF
    FRTPRV=FRTDEF
    DELFPR=DELF
    H=H*LAMBDA
    RADIUS=RADIUS+H
    IF(KOUNT.GT.MAXIT) GOTO 30
C
C
20  KOUNT=KOUNT+1
    CALL FN(GS,PA,NA,NNZ,C,RHO,ABAR,FF,ANGLE,RADIUS,N,RT,CF)
    FRTDEF=FF
C  CHECK FOR CONVERGENCE.....
    IF(ABS(H).LT.1.E-14) GOTO 30
    IF(ABS(FRTDEF).LT.1.E-16) GOTO 30

```

```

C CHECK FOR DIVERGENCE.....
  IF(ABS(FRTDEF).LT.10.0*ABS(FRTPRV)) GOTO 10
  H=H/2.0
  LAMBDA=LAMBDA/2.0
  RADIUS=RADIUS-H
  GOTO 20
C
30  RETURN
    END
C
C
C  -----
C
  SUBROUTINE FN(GS,PA,NA,NNZ,C,RHO,ABAR,F,ANGLE,RADIUS,N,RT,CF)
C
C  -----
C
  INTEGER GS
  INTEGER NA(GS),NNZ,I,C,N
  REAL SUM,PA(GS),F,RADIUS,ABAR,ANGLE,RHO,RN,TOL,PI2
  COMPLEX CSUM,RT,CF
C
  TOL=1.E-14
  PI2=4*ACOS(0.0)
  IF(ABS(RADIUS).LT.1.0E-4) RADIUS=1.0E-4
  SUM=0.0
  CSUM=(0.0,0.0)
  GOTO (100,200,300,400) N
C
C  real version of char. equation
C
100  DO 110 I=1,NNZ
      SUM=SUM+PA(I)*(RADIUS**NA(I))
110  CONTINUE
      F=RADIUS**C-EXP((C*RHO/ABAR)*(SUM-1.0))
      CF=F
      GOTO 500
C
C  complex version of char. equation
C
200  IF(CABS(RT).GT.1.01) THEN
      F=.1E0
      GOTO 500
  ENDIF
  DO 210 I=1,NNZ
      CSUM=CSUM+PA(I)*(RT**NA(I))
210  CONTINUE
      CF=RT**C-CEXP((C*RHO/ABAR)*(CSUM-1.0))
      F=CABS(CF)
      GOTO 500
C
C  modified char. equation
C
300  DO 310 I=1,NNZ
      SUM=SUM+PA(I)*(RADIUS**NA(I))*COS(NA(I)*ANGLE)
310  CONTINUE

```

```

      F=(C*RHO/ABAR)*(SUM-1.0)-C*LOG(RADIUS)
      GOTO 500
C
C      rn value
C
400   DO 410 I=1,NNZ
      SUM=SUM+PA(I)*(RADIUS**NA(I))*SIN(NA(I)*ANGLE)
410   CONTINUE
      RN=((C*RHO/ABAR)*SUM-C*ANGLE)/PI2
      I=RN+TOL
      RN=RN-I
      F=ABS(RN)
C
500   RETURN
      END

```

PROGRAM INPUT

C
C
C
C

```

INTEGER PS,RS,GS
PARAMETER(PS=3000,RS=510,GS=31)
INTEGER C,C2,I,NNZ,NA(GS),MAXIT,SIZE,U
REAL RHO,RRHO,A1,A2,A3,SUM3,SUM4,L1,L2,L3,V1,V2,V3
REAL PA(GS),P(0:PS),ABAR,SUM1,SUM2
REAL RADIUS,ANGLE,RN,F,DIFM,SUM
COMPLEX RT(RS),R,CF

```

C

```

WRITE(*,*) ' WELCOME TO THE QUEUEING SYSTEM MXDC'
WRITE(*,*)
WRITE(*,*)
WRITE(*,*) ' PLEASE ENTER THE # OF DIFFERENT GROUP SIZES'
READ(*,*) NNZ
WRITE(*,*)
WRITE(*,*) ' ENTER THE GROUP SIZE DISTRIBUTION:'
ABAR=0.0
DO 100 I=1,NNZ
    READ(*,*) NA(I),PA(I)
    ABAR=NA(I)*PA(I) + ABAR

```

100

```

CONTINUE
WRITE(*,*)
WRITE(*,*) ' ABAR = ',ABAR
WRITE(*,*)
WRITE(*,*) ' ENTER RHO '
READ(*,*) RHO
WRITE(*,*)
WRITE(*,*) ' ENTER # OF SERVERS (C)'
READ(*,*) C
WRITE(*,*)
WRITE(*,*) ' ENTER DIFM'
READ(*,*) DIFM
WRITE(*,*)
WRITE(*,*) ' ENTER THE MAX # OF ITERATIONS (MAXIT)'
READ(*,*) MAXIT
WRITE(*,*)
WRITE(*,*) '          CALCULATING'

```

C

U=6

C
C
C
C

```

CALL MXDC(NNZ,NA,PA,ABAR,C,RHO,MAXIT,DIFM,RT,P,SIZE)

```

C
C
C
C

```

WRITE(U,*) ' THE ROOTS ARE:'
WRITE(U,*)
WRITE(U,*)
DO 800 I=1,C
    R=RT(I)

```



```

      RADIUS=CABS(R)
      ANGLE=ACOS(REAL(R)/RADIUS)
      CALL FN(GS,PA,NA,NNZ,C,RHO,ABAR,RN,ANGLE,RADIUS,4,R,CF)
      CALL FN(GS,PA,NA,NNZ,C,RHO,ABAR,F,ANGLE,RADIUS,2,R,CF)
      WRITE(U,*) I,' ROOT = ',RT(I)
      WRITE(U,*) '      RN = ',RN
      WRITE(U,*) '      FN = ',F
      WRITE(U,*)
800    CONTINUE
C
C
C
      RRHO=1-RHO
      C2=C*C
      A1=0.0
      A2=0.0
      A3=0.0
      DO 1000 I=1,NNZ
        A1=NA(I)*PA(I)+A1
        IF(NA(I).GT.1) THEN
          A2=NA(I)*(NA(I)-1)*PA(I) + A2
          IF(NA(I).GT.2) THEN
            A3=NA(I)*(NA(I)-1)*(NA(I)-2)*PA(I)+A3
          ENDIF
        ENDIF
1000    CONTINUE
C
      SUM1=0.0
      SUM2=0.0
      SUM3=P(0)*C2
      SUM4=P(0)*C2*C
C
      DO 1100 I=1,C
        IF(I.NE.1) THEN
          SUM1=REAL(1/(1-RT(I))) + SUM1
          SUM2=REAL(RT(I)/((1-RT(I))*(1-RT(I))))+SUM2
        ENDIF
        IF(I.NE.C) THEN
          SUM3=(C2-I*I)*P(I) + SUM3
        ENDIF
        SUM4=(C2*C-I*I*I)*P(I)+SUM4
1100    CONTINUE
C
      L1=SUM1+1/(2*RRHO)*(RHO*A2/A1+1-C*RRHO*RRHO)
      V1=4*RHO*A3*RRHO/A1 + 3*RHO*A2/A1*RHO*A2/A1
      V1=V1+6*RHO*A2/A1*(C*RHO*RHO-2*C*RHO+C-RHO+2) +1+2*RHO
      V1=(V1+6*C*RHO*RRHO*RRHO-C2*RRHO*RRHO*RRHO*RRHO)
      V1=V1/(12*RRHO*RRHO) - SUM2
C
      L2=(C*RHO*(1+A2/A1) - C2*RRHO*RRHO +SUM3)/(2*C*RRHO)
      V2=-3*L2*(C2*RRHO*RRHO+C*(1-2*RHO)-C*RHO*A2/A1)
      V2=V2+3*C*RHO*A2/A1*(-C*RHO+C+1) + C*RHO*A3/A1
      V2=V2+C2*C*RHO*RHO*RHO+3*C*C*RHO*RRHO*(C+1)-C*(C*C-RHO)+SUM4
      V2=V2/(3*C*RRHO) +L2-L2*L2
      L3=0.0
      V3=0.0

```

```

DO 1200 I=0,SIZE-1
  L3=I*P(I)+L3
  V3=I*I*P(I)+V3
1200 CONTINUE
V3=V3-L3*L3

C
C
WRITE(U,*) ' CHECK ON OUTPUT'
WRITE(U,*)
WRITE(U,*) ' USING ROOTS:'
WRITE(U,*) '          MEAN= ',L1
WRITE(U,*) '          VARIANCE= ',V1
WRITE(U,*)
WRITE(U,*) ' USING THE FIRST C PROBS:'
WRITE(U,*) '          MEAN= ',L2
WRITE(U,*) '          VARIANCE= ',V2
WRITE(U,*)
WRITE(U,*) ' USING ALL THE PROBS:'
WRITE(U,*) '          MEAN= ',L3
WRITE(U,*) '          VARIANCE= ',V3
WRITE(U,*)
WRITE(U,*) 'MAXIT = ',MAXIT
WRITE(U,*) 'DIFF = ',DIFM
WRITE(U,*) 'SIZE = ',SIZE-1
WRITE(U,*)
WRITE(U,*) ' THE PROBABILITIES ARE:'
WRITE(U,*)
SUM=0.0
DO 1400 I=0,SIZE-1
WRITE(U,*) ' P(' ,I,') = ',P(I)
SUM=SUM+P(I)
1400 CONTINUE
WRITE(U,*)
WRITE(U,*) ' THE SUM OF THE PROBS = ',SUM
1401 CLOSE(11)
END

```

DISTRIBUTION LIST

Copy No.

- 1 Office of Naval Research
800 North Quincy Street
Arlington, VA 22217
- Attention: Scientific Officer,
Statistics and Probability
Mathematical Sciences Division
- 2 ONR Resident Representative
Joseph Henry Building, Room 623
2100 Pennsylvania Avenue, N.W.
Washington, DC 20037
- 3 - 8 Director, Naval Research Laboratory
Washington, DC 20375
- Attention: Code 2627
- 9 - 20 Defense Technical Information Center
Building 5, Cameron Station
Alexandria, VA 22314
- 21 - 29 C. M. Harris
- 30 GMU Office of Research