

AD 744802

AD

PAA-TRI-72

THE NUMERICAL SOLUTION OF TRANSIENT QUEUEING PROBLEMS



TECHNICAL REPORT

BY

STUART W. OLSON

RECEIVED
JUL 19 1972
C

SYSTEMS ANALYSIS DIVISION
PLANS AND ANALYSIS DIRECTORATE
HQ, US ARMY WEAPONS COMMAND
ROCK ISLAND, ILLINOIS

DISTRIBUTION

Approved for publication
Dist. But not to be published

NOTAL
FILE

127

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Systems Analysis Division Plans and Analysis Directorate HQ, US Army Weapons Command		2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	
		2b. GROUP	
3. REPORT TITLE THE NUMERICAL SOLUTION OF TRANSIENT QUEUEING PROBLEMS			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Final Report			
5. AUTHOR(S) (First name, middle initial, last name) Stuart W. Olson			
6. REPORT DATE May 1972		7a. TOTAL NO. OF PAGES	7b. NO. OF REFS 34
8a. CONTRACT OR GRANT NO.		9a. ORIGINATOR'S REPORT NUMBER(S) PAA-TRI-72	
b. PROJECT NO.			
c.		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.			
10. DISTRIBUTION STATEMENT Distribution of this document is unlimited			
11. SUPPLEMENTARY NOTES Master's Thesis		12. SPONSORING MILITARY ACTIVITY HQ, US Army Weapons Command Rock Island, Illinois 61201	
13. ABSTRACT This report explores methods for obtaining transient solutions to queueing problems which can be represented in the form of differential-difference equations. Six distinct methods, representing the most frequently-encountered in the open literature, are discussed as to their value in numerical work. The method of Runge-Kutta integration of these equations was found to be superior to the numerical evaluation of analytic solutions of a particular queueing model. A generalized, Runge-Kutta programming package, written in FORTRAN IV for the IBM 360/65, is presented and described in detail for use on queueing problems. Generalization is achieved by requiring the user to write a subroutine to evaluate his queueing equations when required by the programming package. Several sample problems are presented and solved to demonstrate the wide potential applicability of this method and associated computer program. The advantage of this method is that the usual rate parameters to describe state transitions may be any functions freely-chosen by the user. The appendix which contains the FORTRAN program and instructions for its use may be separated from the report and used separately.			

DD FORM 1473
1 NOV 65REPLACES DD FORM 1473, 1 JAN 64, WHICH IS
OBSOLETE FOR ARMY USE.

UNCLASSIFIED

Security Classification

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Queueing Theory						
Transient Solution Methods						
Numerical Methods						
Computers						
Programming						
FORTRAN						

UNCLASSIFIED

Security Classification

THE NUMERICAL SOLUTION OF TRANSIENT
QUEUEING PROBLEMS

By

Stuart W. Olson

Systems Analysis Division
Plans and Analysis Directorate
HQ, US Army Weapons Command
Rock Island, Illinois

May 1972

PAA-TR1-72

ABSTRACT

This report explores methods for obtaining transient solutions to queueing problems which can be represented in the form of differential-difference equations. Six distinct methods, representing the most frequently-encountered in the open literature, are discussed as to their value in numerical work. The method of Runge-Kutta integration of these equations was found to be superior to the numerical evaluation of analytic solutions of a particular queueing model.

A generalized, Runge-Kutta programming package, written in FORTRAN IV for the IBM 360/65, is presented and described in detail for use on queueing problems. Generality is achieved by requiring the user to write a subroutine to evaluate his queueing equations when required by the programming package. Several sample problems are presented and solved to demonstrate the wide potential applicability of this method and associated computer program. The advantage of this method is that the usual rate parameters to describe state transitions may be any functions freely-chosen by the user.

The appendix which contains the FORTRAN program and instructions for its use may be separated from the report and used separately.

ACKNOWLEDGEMENTS

This report is identical to the author's Master's thesis for the Department of Management and Industrial Engineering at the University of Iowa. The author is deeply indebted to the Army Weapons Command which provided support during the study program and particularly during the preparation of the thesis.

The author wishes to gratefully acknowledge his thesis supervisor Professor Liittschwager for his valuable stimulation and guidance during the preparation of this thesis and for his foresight in suggesting this topic for study.

The author also acknowledges the earlier stimulation and guidance of Mr. George Schlenker who originally suggested the approach to the problem.

The author greatly appreciates the services of Sandi Conley, who typed the manuscript.

TABLE OF CONTENTS

	Page
LIST OF FIGURES	v
LIST OF TABLES	vi
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 REVIEW OF METHODS FOR	
PARTICULAR TRANSIENT QUEUES	3
Generating Functions	8
Analytic Continuation	10
Power Series Expansions	11
Integral Equations	13
Monte-Carlo Simulation	17
Runge-Kutta Integration	18
CHAPTER 3 A COMPARISON OF THREE METHODS	
FOR THE M/M/1 QUEUE	23
Analytic Bessel Function Method	23
Runge-Kutta Method	25
Monte-Carlo Simulation	32
CHAPTER 4 EXAMPLES OF SOLUTIONS TO OTHER	
QUEUEING MODELS	34
Preemptive-Resume Queue	34
State-Dependent Queue	46

CHAPTER 5 SUMMARY	55
LIST OF REFERENCES	58
GENERAL REFERENCES	60
APPENDIX A LISTING OF MONTE-CARLO SIMULATION PROGRAM	62
APPENDIX B LISTING OF BESSEL FUNCTION PROGRAM	75
APPENDIX C RUNGE-KUTTA PROGRAM	88
Solution of the M/M/1 Queue	93
Subroutine INIT	94
PRIME Entry Point	97
NEWVAL Entry Point	98
WRAPUP Entry Point	99
Dimensioning the Problem	113
Program Output	116
Debugging Aids	117
Job Control Cards	118

LIST OF FIGURES

Figure		Page
1	CPU Time Versus Number in the System N for Bessel Function Method	27
2	CPU Time Versus Number in the System N for Runge-Kutta Method	28
3	Premptive-Resume M/M/1 $\lambda=0.067$, $\mu=0.333$ $\lambda_p=0.00167$, $\mu_p=0.0333$ Expected Number and Standard Deviation of Regular and Priority Systems	45
4	1-channel, 2-servers with $\rho=1.5$, $0 < \tau < 8$ then $\rho=0.5$, $\tau > 8$ for $b=c=0$	54

LIST OF TABLES

Table		Page
1	Comparative CPU Times	26
2	Summary of Benefits	30
3	Program Logic	92

CHAPTER 1

INTRODUCTION

This thesis explores problems associated with obtaining the transient solutions of queueing problems with varying rate parameters and various solution methods that have been employed. The work of Bhat (2) and Neuts (11) has shown that the lack of numerical work in the solution of transient queueing models has "handicapped practitioners to obtain meaningful results" (2). This thesis addresses that problem by presenting a complete, well-tested computer programming package for obtaining Runge-Kutta solutions to transient queueing problems. The thesis concludes that Runge-Kutta integration of the differential-difference equations of queueing problems is "best" in several respects, given the current state-of-the-art of available computational facilities.

Chapter two deals with a review of the computational aspects of several techniques contained in the literature. These include closed-form analytic solutions, approximations, simulation and Runge-Kutta integration. In Chapter three, an evaluation of three basic methods is made on the basis of computer programs which were written, tested and analyzed for each. This Chapter utilizes the single-channel, single-server queue as the principle comparative

model. Examples of solutions to other queueing models are then presented in Chapter four. The thesis is then summarized in Chapter five, followed by Appendicies A, B and C which present listings of computer programs used in Chapters three and four.

The programming package for Runge-Kutta solution of queueing problems is presented in detail in Appendix C. Included with it are complete instructions to guide users in its application to their problems and a detailed explanation of a sample problem.

CHAPTER 2

REVIEW OF METHODS FOR PARTICULAR TRANSIENT QUEUES

Queueing problems are frequently modeled in the form of a set of simultaneous, first-order, differential equations called the Chapman-Kolmogorov (5) equations. These are

$$\dot{\underline{P}}(t) = \underline{A} \underline{P}(t),$$

where $\dot{\underline{P}}(t)$ and $\underline{P}(t)$ are column vectors and $A = \{a_{ij}\}$ is a system transition matrix representing the time-rate of a transition from state i to state j . If $\underline{P}(t) = [P_0(t), P_1(t), \dots, P_N(t)]$, then $P_n(t)$ represents the probability that the system is in state n at time t . Correspondingly, $\dot{P}_n(t)$ represents the time rate of change of $P_n(t)$. In queueing, the system is usually thought of as a set of channels and servers, related by a queue discipline which describes the flow of items in the system. Thus the notation $\dot{P}_n(t)$ describes the time fluctuation of the probability of seeing n items in the system for $0 \leq n < \infty$.

For the M/M/1 queue (15), the general equation for $n \geq 1$ in the system is

$$\dot{P}_n(t) = \lambda P_{n-1}(t) - (\lambda + \mu) P_n(t) + \mu P_{n+1}(t).$$

The resulting matrix form, limiting the maximum number in the system to N is

$$\begin{bmatrix} \dot{P}_0(t) \\ \dot{P}_1(t) \\ \dot{P}_2(t) \\ \dots \\ \dot{P}_N(t) \end{bmatrix} = \begin{bmatrix} -\lambda & \mu & 0 & \dots & 0 & 0 \\ \lambda & -(\lambda+\mu) & \mu & \dots & 0 & 0 \\ 0 & \lambda & -(\lambda+\mu) & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & -(\lambda+\mu) & 0 \end{bmatrix} \begin{bmatrix} P_0(t) \\ P_1(t) \\ P_2(t) \\ \dots \\ P_N(t) \end{bmatrix}.$$

In the transient queueing problems considered by this thesis it is desired to admit rate parameters a_{ij} of any arbitrary form and functional dependence on either time t or number in the system n . Many analytic techniques require that n tend to infinity for a solution. In this thesis, the Runge-Kutta methods discussed will require n be definitely bounded at some finite value N . Further consideration regarding this point will be presented in Appendix C.

All too frequently, only steady-state results are considered, i.e., the solution of

$$\underline{0} = A \underline{P}$$

even when the system under study has frequent inherent changes in arrival or service rates. Numerous situations can be found where arrival rates, service rates and perhaps other system parameters are changing continually. One such situation is in a repair shop. Initially, the shop is empty at the start of the day. Arrivals occur at a rate greater than the service rate during the day. Steady state solutions to such systems are inadequate. What is needed is a method which will yield transient solutions to such problems with no more difficulty than that required for steady state solutions. That method is presented in this thesis.

The argument for seeking a standard technique for solving a wide class of transient queueing problems has a strong practical appeal. The technique can be studied on problems with known solutions in order to determine its properties under wide ranging conditions. The analyst can thus develop intuitive insights which are helpful when applying the technique to problems whose solutions are not known. In applying a standard solution technique, however, the analyst runs the risk of overlooking a possible analytic solution: had he tried some other methods. In Chapter three the economics of attempting to solve the analytic closed form of a transient queueing problem versus using Runge-Kutta and Monte-Carlo techniques will be discussed further.

A search of the literature reveals that transient solutions to numerous queueing problems can be found.

However, in most instances these solutions appear in graphical form with no reference to the numerical techniques used to obtain them. In cases where the numerical techniques employed are revealed (12, 13) little discussion has dealt with the reason for selecting the particular technique from among the many available. Therefore, the literature concerning explicit techniques for transient queues is quite limited although at least eight techniques have appeared in the open literature. Six of the most useful are

1. Use of generating functions (z-transforms)
2. Analytic continuation of the differential equations
3. Series expansion of rate parameters
4. Transformation to integral equations
5. Monte-Carlo simulation
6. Direct integration of the differential equations

Two additional methods do not appear promising for study here. These are the method of numerical inversion of the Laplace transform and the method of eigenvectors. The method of numerical inversion does not offer the desired qualities of generality and numerical stability. The Laplace transform is difficult to obtain when rate parameters are functions of time. Also, in a thesis by Schlenker (16, p.14) the numerical method proposed by Bellman is shown to be frequently unstable and limited in accuracy.

The method of eigenvectors also does not seem promising. In a paper by Kaspar (8), the method is

discussed on a multiple, parallel channel system with no queues allowed. Kaspar claims that if the A matrix can be shown to have distinct eigenvalues, then a new system of linear differential equations with separated variables can be obtained. In our problem, however, the matrix A may be time-varying in such a way as to cause this method to fail. Therefore this method is of no further interest in this thesis.

Many of these methods have been reviewed by Leese and Boyd (10). They recommend an integral equation technique as the best method to solve the M/M/1 queueing problem based on computational time and storage considerations. This chapter draws upon their review and computational experience.

The six more useful methods are now discussed in greater detail. In the first four, the model to be used is the M/M/1 queue with equations written as

$$\dot{P}_0(t) = -\rho(t)P_0(t) + P_1(t) \quad , \text{ and for } n > 1$$

$$\dot{P}_n(t) = \rho(t)P_{n-1}(t) - (\rho(t)+1)P_n(t) + P_{n+1}(t) \quad ,$$

where $\rho(t) = \lambda(t)/\mu$ and t is measured in units of μ .

Generating Functions

In this section, transient solutions obtained by application of generating functions are discussed. Using the equations for $P_n(t)$ as defined above, the following generating function

$$F(x,t) = \sum_{n=0}^{\infty} x^n P_n(t)$$

may be defined. Partial differentiation with respect to x and t and elimination of $P_n(t)$ terms yields the Fokker-Planck Diffusion equation

$$x(1-x) \frac{\partial^2 F}{\partial x \partial t} - (1-x)^2(1-\rho x) \frac{\partial F}{\partial x} + \frac{\partial F}{\partial t} + \rho(1-x)^2 F = 0.$$

Solutions to this equation with finite difference techniques are frequently difficult to obtain due to instabilities. Other techniques, such as the method of characteristics and the particle-in-cell method have proven unsuccessful in various degrees when $\rho(t)$ is nonlinear with respect to t .

This technique has only thus succeeded in transforming a difficult problem into an awkward problem for the numerical analyst. Computationally, partial differential equations also require large amounts of core storage, making them expensive to solve as well. Also, the method makes explicit use of the differential equations which means that

for each model, a new derivation for the partial differential equation is required.

The use of a generalized generating function of the form

$$F(x,t) = \sum_{n=0}^{\infty} f_n(x,t) P_n(t)$$

can be sometimes used to advantage when a particular choice of f_n will simplify the resulting partial differential equation. For example, if f_n is a polynomial in x with degree n and $P_n(t)$ is as above, $F(x,t)$ will satisfy

$$\frac{\partial F}{\partial t} = \rho(x-1)F$$

if x satisfies

$$\rho f_{n+1} + f_n + f_{n-1} = (\rho x + 1) f_n, \quad n \geq 1$$

with

$$f_0 = 1 \quad \text{and} \quad f_1 = x.$$

In this case the second order partial differential equation is thus simplified to a first order system with an auxiliary algebraic condition.

Another means of solving partial differential equations is by the method of integral equations. This method, sufficiently important in its own right, is discussed in a separate section of this chapter starting on page 13.

Analytic Continuation

A discussion is now given of the method of analytic continuation of the function $P_n(t)$ in the interval $(t, t+h)$ using Taylor's expansion.

Knowing $P_n(t)$ for some t allows one to obtain $\dot{P}_n(t)$ for all n . Repeated differentiation of the equations $P_n(t)$ yields

$$\frac{d^{(j)}}{dt^{(j)}} [P_n(t)] .$$

Applying Taylor's formula, the expression at $t+h$ becomes

$$P_n(t+h) = \sum_{j=0}^{\infty} \frac{h^j}{j!} \frac{d^{(j)}}{dt^{(j)}} [P_n(t)] .$$

At least eight terms of the series are usually required for meaningful accuracy. We need only note that if ρ is a function of t , the chain rule of differentiation will make calculation of the derivatives rather tedious. Although programs for doing this (FORMAC, SNOBOL, LISP, etc.) are available, they are expensive and difficult to use.

The use of this technique requires an error analysis of the contribution of truncated terms of the series. Usually, an attempt is made to bound the error with an expression involving the number of terms before truncation, h , and the magnitude of higher derivatives of $P_n(t)$ (1).

Power Series Expansions

This technique uses the Bessel function solution which is apparently originally due to Clark (3, 15). This solution is given here in Chapter three on page 24. The Bessel functions and exponential terms of this solution are rearranged such that $P_n(t)$ is expressed as a power series in ρ of the form

$$P_{N,n}(t) = \sum_{m=0}^{\infty} \rho^m C_{N,n}^m(t)$$

where

$$C_{N,n}^m(t) = T_m(t) \nabla^m S_{N+m-n}(t) - T_{m-n}(t) \nabla^{m-n-1} S_{N+m}(t) \\ + T_{m-n-1}(t) \nabla^{m-n-2} S_{N+m}(t) ,$$

$$T_j(t) = \frac{t^j}{j!} \quad (j \geq 0), \quad T_j(t) = 0 \quad (j < 0) ,$$

$$S_j(t) = e^{-t} T_j(t), \quad \text{and}$$

$$-\nabla^{-1} S_{j-1}(t) = \sum_{u=j}^{\infty} S_u(t) .$$

This method works well if ρ can be approximated by constant values over short, equally-spaced intervals, since the C 's need only be computed once and stored for all N, m, n and t . However, a large amount of computing time and high-speed storage are required. Also, the method is not exact and requires a great deal of effort to establish the expressions for the C 's if the form of the equation changes. The problem of numerical instability in evaluating the terms in the infinite sums must be constantly considered by the analyst, since the C 's can exceed unity.

To avoid this problem, the terms in the power series expansion can be written differently leaving the Poisson part unaltered. Thus

$$P_{N,n}(t) = \sum_{m=0}^{\infty} \left[\frac{(\rho t)^m e^{-\rho t}}{m!} \right] F_{N,n}^m(t)$$

where

$$F_{N,n}^m(t) = S_{m+N-n}(t) + \frac{T_{m-n}(t) - T_{m-n-1}(t)}{T_m(t)} \sum_{u=m+N+1}^{\infty} S_u(t).$$

In this form, each of the terms $[.]$ and F represent probabilities and are thus less than unity. Although the possible computational difficulties are largely avoided with this new form, obtaining a new set of equations for each different queueing problem encountered would be tedious.

Integral Equations

In this section, techniques will be discussed for solving partial differential equations such as the form of the Fokker-Planck equation developed in the section under Generating Functions. A solution to the Fokker-Planck partial differential equation may be formulated in terms of a Volterra-type integral equation; however it is not suitable for numerical work.

One formulation, depending upon a change of variable, has been found quite suitable for solution by computer, and in fact, is probably the most efficient in terms of speed and storage required. This is the method of Wragg as implemented by Leese and Boyd (10). The change of variable is

$$q_n(t) = \sum_{j=n}^{\infty} P_j(t) .$$

Then by substitution into the equations for $P_n(t)$

$$\dot{q}_n(t) = \rho q_{n-1}(t) - (1+\rho)q_n(t) + q_{n+1}(t) , \quad n \geq 1$$

with $q_0(t) = 1$ and $q_n(0) = 0$, $n \geq 1$.

By defining

$$u = \int_x^t \rho(y) dy , \quad v = \left[\frac{u}{t-x} \right]^{1/2} \text{ and } \omega = 2 u(t-x)^{1/2}$$

and writing in original $q_n(t)$ form, we obtain

$$q_n(t) = \int_0^t [I_{n-1}(\omega) \rho(x) v^{n-1} - I_{n+1}(\omega) v^{n+1}] \exp(-u-t+x) G(x) dx.$$

In this expression $I_n(\omega)$ is a Bessel function of the First Kind and $G(x)$ satisfies

$$1 = G(t) + \int_0^t K(t,x) G(x) dx ,$$

where

$$K(t,x) = \left[\frac{\rho(x)}{v} - v \right] I_1(\omega) \exp(-u-t+x) .$$

This technique still involves solution of Bessel functions, complicated by the fact that solution for $G(t)$ requires iteration. The above integral equation is far removed from the original model in terms of analytical transformations. To arrive at it, the form and assumptions of the original queueing problem were heavily used. Thus, altho this is an impressive method of solving for the transient conditions in the M/M/1 model, it can hardly be relied upon as a technique for a broad class of problems.

One other technique using partial differential equations for solution is useful in large-scale queueing systems. This method is due to Newell (12) who applied it in highway traffic problems.

Define $F(x,t) = P[X(t) < x]$ where $X(t)$ is the queue length at time t , then the Fokker-Planck equation may be used as a second-order approximation of $\frac{\partial F(x,t)}{\partial t}$ where

$$\frac{\partial F}{\partial t} = a(t) \frac{\partial F}{\partial x} + \frac{b(t)}{2} \frac{\partial^2 F}{\partial x^2}.$$

This is valid for large x , a large number of arrivals and departures in time interval τ with only a small change of x during τ and further, that the number of arrivals and departures during τ is normally distributed. These conditions are often not unduly restrictive for large scale systems.

Let $E[A(t)]$ be the number of arrivals until t and $E[D(t)]$ be the number of departures until t .
Then

$$\begin{aligned} \tau a(t) &= E[A(t) - D(t)] - E[A(t+\tau) - D(t+\tau)] \\ &= E[D(t+\tau) - D(t) + A(t+\tau) - A(t)] \end{aligned}$$

and similarly

$$\tau b(t) = \text{VAR}[D(t+\tau) - D(t) + A(t+\tau) - A(t)].$$

By defining $t^* = \frac{t}{T_0}$ and $x^* = \frac{x}{L_0}$ then

$$\frac{\partial F}{\partial t^*}(x^*, t^*) = \frac{Ta(t^*)}{L} \frac{\partial F}{\partial x^*}(x^*, t^*) + \frac{Tb(t^*)}{2L^2} \frac{\partial^2 F(x^*, t^*)}{\partial x^{*2}}$$

Now if $T = T_0$ and $L = L_0$ at $x=0$, then

$$\frac{\partial F}{\partial t^*} = \frac{T_0}{L_0} a_0 \frac{\partial F}{\partial x^*} + \frac{T_0 b_0}{2L_0^2} \frac{\partial^2 F}{\partial x^{*2}} .$$

Choose b_0 such that $\frac{T_0 b_0}{2L_0^2} = \frac{1}{2}$ and

let $a(t) = -\alpha t$ so that

$$\frac{\partial F}{\partial t^*} = -\alpha t^* \frac{\partial F}{\partial x^*} + \frac{1}{2} \frac{\partial^2 F}{\partial x^{*2}} .$$

This equation is solved using a simple finite difference analogue similar to a random walk on a lattice. By letting

$$X_{j+1} = \begin{cases} X_j + 1 & \text{with probability } p \\ \max(X_j - 1, 0) & \text{with probability } (1-p) \end{cases}$$

where

$$p_j = \frac{1}{2} (1 + \alpha_j)$$

and with $F_j(k) = P\{X_j \leq k\}$, then

$$F_{j+1}(k) = p_j F_j(k-1) + (1-p_j) F_j(k+1), \quad k \geq 1$$

and $F_{j+1}(0) = (1-p_j)F_j(1)$, $\alpha_j \ll 1$,

which can be applied iteratively to solve

$$\frac{\partial F_j(k)}{\partial j} = (1-\alpha_j) \frac{\partial F_j(k)}{\partial k} + \frac{1}{2} \frac{\partial^2 F_j(k)}{\partial k^2} .$$

Monte-Carlo Simulation

Monte-Carlo simulation is a powerful tool for solving queueing problems in the sense that it can presumably be applied to any model. In cases where only a narrative description of the problem exists or where differential equations can not be derived describing the model, Monte-Carlo is perhaps the only quantitative approach. The relevant question then is how efficient is Monte-Carlo as compared to numerical techniques? Modern computational equipment, simulation languages, variance-reduction techniques and refined cost accounting of computer programs have made Monte-Carlo simulation an attractive alternative for an analyst who may not wish to spend a great deal of his time attempting to derive a set of differential equations to describe his system. Monte-Carlo simulation of a simple model is used when it is contemplated that the model will be enriched with complexities that will make it analytically intractable at a later time.

Monte-Carlo simulation is a reasonable way of estimating the steady-state solution of a system. Estimating transient behavior, however, is much more difficult and costly if it is desired to estimate $P(t)$ at many discrete values of t . Since, in reality, analysts are generally interested in obtaining quantities such as the waiting time, number in the system, and the number of units processed, $P(t)$ is not required and the desired quantities may be observed directly and economically.

A simulation of a $M/M/1$ queue was written in SIMSCRIPT-II (J) for this thesis in order to form a basis for comparison with other methods. SIMSCRIPT-II is probably one of the most suitable and efficient languages available for simulation involving transient queueing behavior. The simulation and results are given in the next chapter.

Runge-Kutta Integration

Direct solution of the differential equations by Runge-Kutta methods is not only simple and straightforward, but well understood, flexible and easy to apply to new problems. The numerical analysis and behavior of Runge-Kutta are well-known. The method is the most stable of the methods commonly used in differential equations (Euler, Milne, Adams Bashforth, etc.), it is easy to use and produces error of order h^4 , where h is the step size. This author has

written a subroutine which will integrate simultaneous differential equations of arbitrary size and order and has established several years of experience with it on wide classes of problems encountered in differential equations. The Chapman-Kolmogorov equations are a particularly "worry-free" class of problems solved by the Runge-Kutta method. Therefore, it seems reasonable to attempt the development of a general computer program to solve these equations. To a large extent, this has been accomplished.

The author's program consists of unit modules with multiple entry points. The user writes a subroutine named INIT with only the minimal amount of FORTRAN coding necessary to describe his equations and problem peculiarities. The first entry point is used for initialization. The second entry point is used to provide values for $\dot{P}_n(t)$, given $P_n(t)$ and t . The third entry point is used after each step in the solution to allow for computation of auxiliary quantities of interest such as queue statistics. The fourth entry point is used at the completion of the solution to re-initialize for another problem or to go on to some other task. These entry points will be discussed in greater detail in Appendix C. The programming of subroutine entry points is discussed in (6) in detail.

The Runge-Kutta subroutine, as described below, has been adapted following the form given by Nielsen in (14, p.

178). In this form, which is one of many alternative forms in the literature, it is possible to solve a set of n -th order simultaneous differential equations without explicitly redefining the higher order derivatives to obtain the usual first order system.

The Runge-Kutta subroutine performs four steps which involve evaluation of the differential equations to move the solution for $P_n(t)$ from t to $t+\Delta t$. Initially set

$$\Delta P_i = \dot{P}_i(t) \quad i=1,N.$$

Then the four Runge-Kutta steps appear as below.

R-K

Step No. Operations Performed

$$k=1 \quad t_1 = t + \frac{\Delta t}{2} \quad P_{i1} = P_i + \dot{P}_i \frac{\Delta t}{2}; \quad i=1,N$$

$$\dot{P}_{i1} = \dot{P}_i(t_1); \quad i=1,N \quad \Delta P_i + \Delta P_i + 2\dot{P}_{i1}; \quad i=1,N$$

$$k=2 \quad t_2 = t + \frac{\Delta t}{2} \quad P_{i2} = P_i + \dot{P}_{i1} \frac{\Delta t}{2}; \quad i=1,N$$

$$\dot{P}_{i2} = \dot{P}_i(t_2); \quad i=1,N \quad \Delta P_i + \Delta P_i + 2\dot{P}_{i2}; \quad i=1,N$$

$$k=3 \quad t_3 = t + \Delta t \quad P_{i3} = P_i + \dot{P}_{i2} \Delta t; \quad i=1,N$$

$$\dot{P}_{i3} = \dot{P}_i(t_3) ; i=1,N \quad \Delta P_i \leftarrow \frac{\Delta t}{6} (\Delta P_i + \dot{P}_{i3}) ; i=1,N$$

$$k=4 \quad t_4 = t + \Delta t \quad P_{i4} = P_i + \Delta P_i ; i=1,N$$

$$\dot{P}_{i4} = \dot{P}_i(t_4) ; i=1,N$$

$$t = t_4 \quad P = P_{i4} ; i=1,N \quad \dot{P}_i = \dot{P}_{i4} ; i=1,N$$

It should be noted that only three working storage vectors of length N need be provided for the P_i, P_{ik} and $\dot{P}_{ik}$ values. These are provided by the user's subroutine and may be used for other storage requirements when not in use by the four-step Runge-Kutta computations.

Rules for choosing Δt have been developed to allow it to change as the solution progresses, depending on the magnitude of some measure of truncation error (1). Generally, these rules are heuristic or highly problem-dependent. This author recommends a constant Δt initially and then re-running the problem at a few selected values for Δt , which should give the user a good feel for the trade-off between computational time (cost) versus quadrature error. Since the P_i values must sum to unity for any and all t , the difference between this sum and unity makes an excellent measure of accumulated error at each step of the solution.

Further work on this problem should consider the optimality of the weights of the $\dot{P}_{ik}$ in computing the

ΔP_i above. Currently,

$$\Delta P_i = \Delta t (a \dot{P}_i + b \dot{P}_{i1} + c \dot{P}_{i2} + d \dot{P}_{i3}) / w$$

where $a=1, b=2, c=2, d=1$ and $w=a+b+c+d$. These values are optimal if the function being integrated is a parabola on the interval from t to $t+\Delta t$. If one were to replace this assumption by the fact that the P_i must always sum to unity, it might be possible to derive a new set of weights more specifically suited to this kind of problem.

The next chapter will present a detailed comparison of a method representative of those discussed in this chapter, the Runge-Kutta method and Monte-Carlo simulation.

CHAPTER 3

A COMPARISON OF THREE METHODS FOR THE M/M/1 QUEUE

In this chapter a comparison is made between a representative analytic solution, the Runge-Kutta solution of a single-channel, single-server queue, and the Monte-Carlo method. The analytic solution for this model is presented in Saaty (15) and is considered representative in complexity and computational difficulty of all the analytical methods discussed in the previous chapter. The comparison of methods will consist of computing $\dot{P}(t)$ from $t=0$ until steady state is achieved, i.e., $t=t^*$ where $\dot{P}(t^*) \approx 0$.

Analytic Bessel Function Method

The time-dependent solution of the single-channel, single-server queue is derived (apparently by Clarke (3)) in terms of modified Bessel functions of the First Kind.

This is given by Saaty as

$$\begin{aligned}
 P_n(t) = & \exp[-(\lambda+\mu)t] \left\{ (\lambda/\mu)^{\frac{n-i}{2}} I_{n-i}(2\sqrt{\lambda\mu} t) \right. \\
 & + (1-\lambda/\mu)(\lambda/\mu)^n \sum_{k=n+i+2}^{\infty} (\lambda/\mu)^{\frac{-k}{2}} I_k(2\sqrt{\lambda\mu} t) \\
 & \left. + (\lambda/\mu)^{\frac{n-i+1}{2}} I_{n+i+1}(2\sqrt{\lambda\mu} t) \right\},
 \end{aligned}$$

where i is the initial number of items in the system. A computer program was written in FORTRAN using the Bessel function subroutine BESI from the IBM Scientific Subroutine Package (SSP) (7). The listing of this program with sample output is included in Appendix B. The size of the load module was 29,856 bytes. Time t was varied from 0 to 20 in steps of 0.1 and n varied from 0 to 20 with $i=0$. The parameters λ and μ were chosen as 0.45 and 0.5, respectively. The equation for $P_n(t)$ above was solved for 201 values of t and 21 values of n for a total of 4221 times. The amount of central processor unit (CPU) time for the IBM 360/65 was 98.64 seconds. Approximately four hours were required to write the program and several trials were needed to debug it. The infinite series term of the equation was truncated when either $k=n+50$ or the k -th term was less than 10^{-6} . This rule produced values of $P_n(t)$ which were no more than about 1% in error.

Runge-Kutta Method

The single-channel, single-server equations were integrated using the Runge-Kutta method. For completeness they are listed here again as

$$\dot{P}_0(t) = -\lambda P_0(t) + \mu P_1(t)$$

$$\dot{P}_n(t) = \lambda P_{n-1}(t) - (\lambda + \mu) P_n(t) + \mu P_{n+1}(t) \text{ for } 1 \leq n \leq N$$

$$\dot{P}_N(t) = \lambda P_{N-1}(t) - (\lambda + \mu) P_N(t) \quad \text{with} \quad P_0(0) = 1.$$

The same values for t, λ and μ were used with $N=20$. The time step Δt was chosen as 0.1. Since the Runge-Kutta method requires four evaluations of the equations per step, the above equations were evaluated a total of $201 \times 4 = 804$ times for each value of n . This is $804 \times 21 = 16,884$ equations evaluated for the run. The Runge-Kutta method requires storage of the entire $\underline{P}(t)$ and $\dot{\underline{P}}(t)$ vectors plus three more working vectors of the same length. The size of the load module for this program was 25,960 bytes. The amount of CPU time for the solution was 7.42 seconds. The solution error grew to a maximum of 0.005% with this step size. Only the differential equations and initial conditions had to be programmed as a subroutine named INIT since the main program is completely general. This consumed about 1/2 hour and did not need to be debugged.

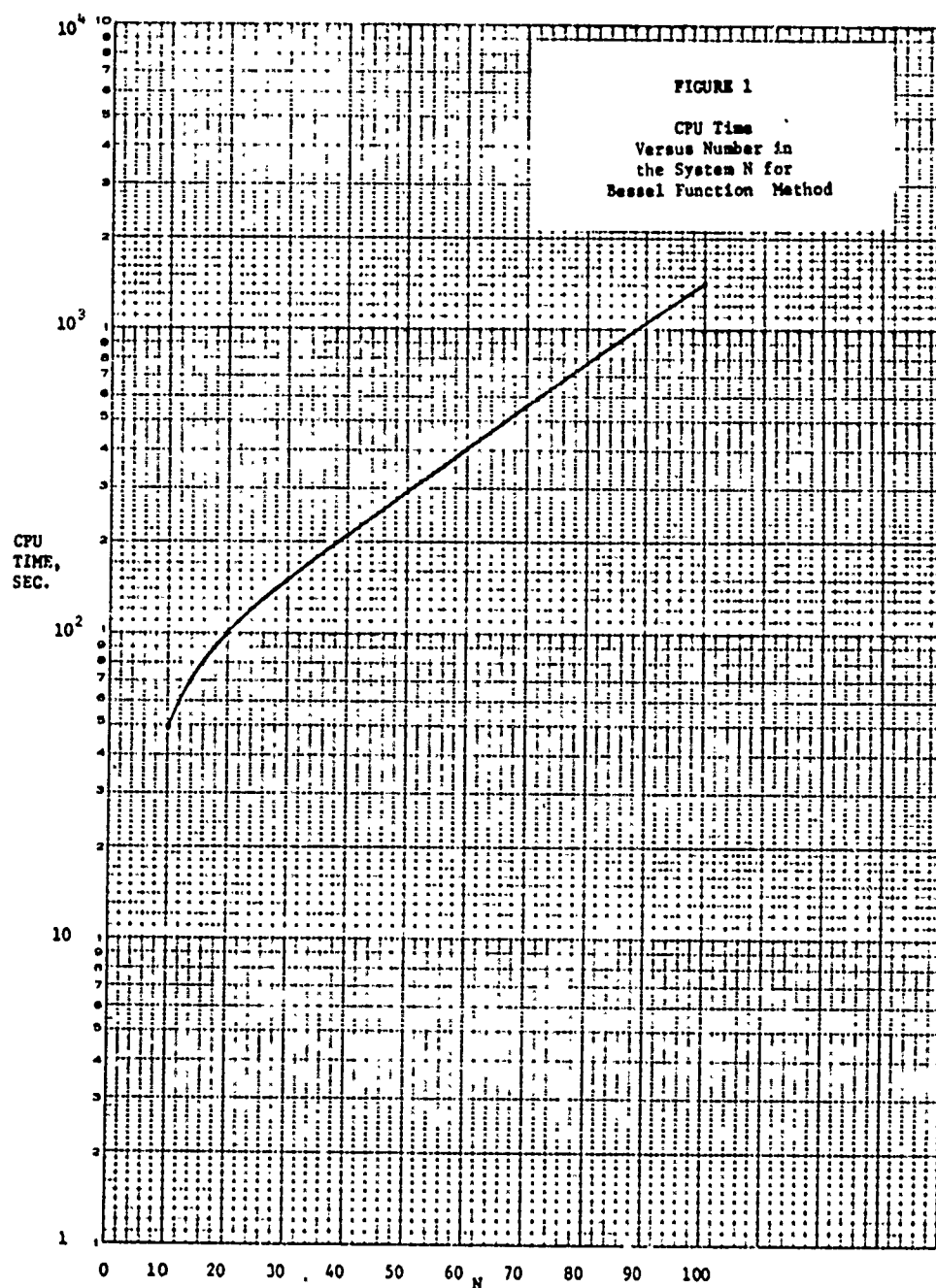
The following table illustrates how CPU time varies as a function of the maximum number in the system N for the Bessel function solution and for Runge-Kutta solution. Times are in CPU seconds.

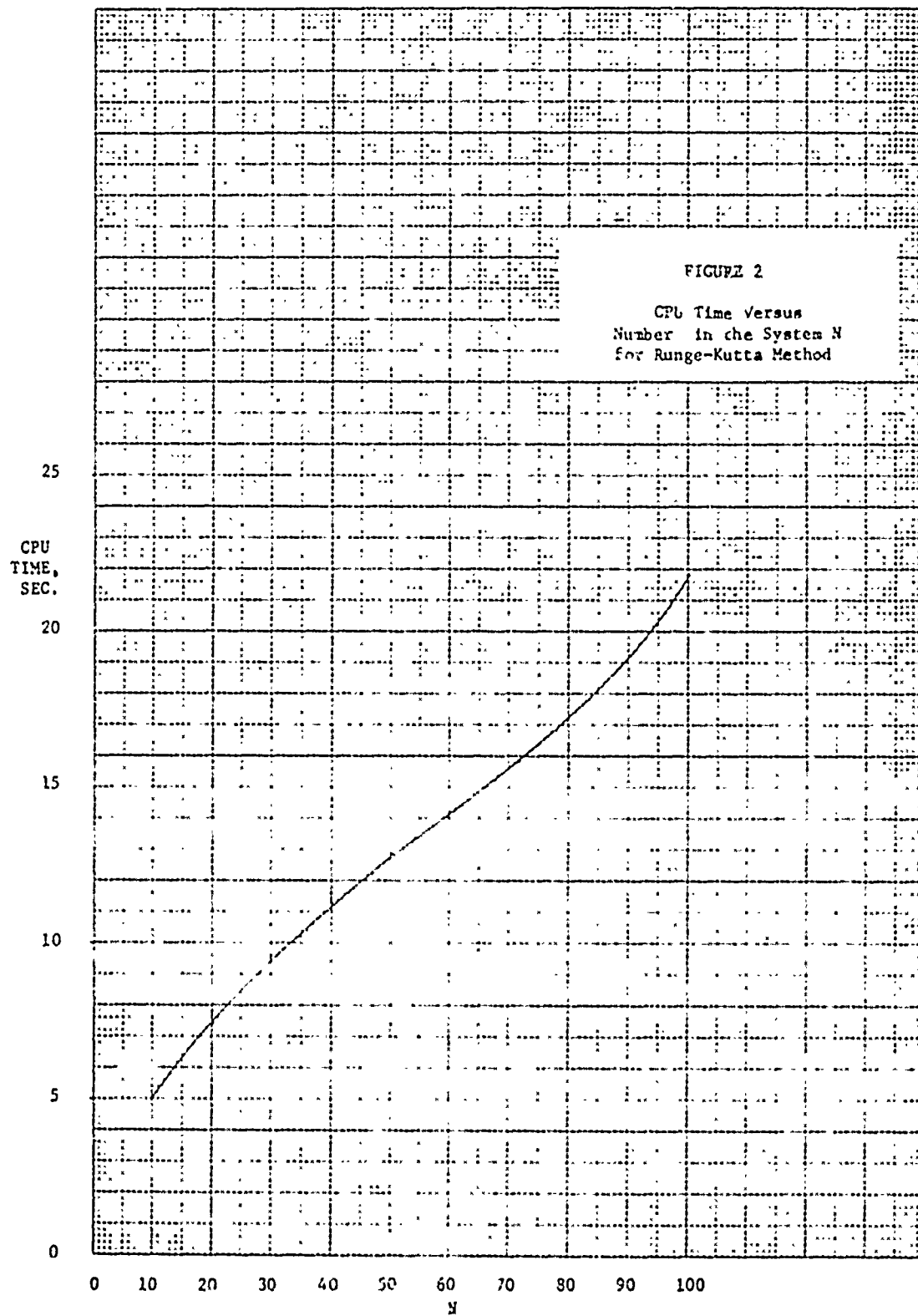
Table 1
Comparative CPU Times

<u>N</u>	<u>Bessel Functions</u>	<u>Runge-Kutta</u>
10	59.0	5.0
20	100.9	7.4
30	149.9	9.6
40	205.2	11.3
50	290.6	12.9
75	677.9*	16.4
100	1428.5*	21.8

The asterisk (*) notation above indicates these figures are linear extrapolations based upon runs over a shorter range of t , since there is a uniform correspondence between CPU time for the run and t .

When plotted, the times for the Bessel function solution vary as a power function with N (Figure 1). The Runge-Kutta solutions vary linearly with N (Figure 2).





Estimates of required CPU time T may be obtained from the empirical relations

$$T = 46 \exp(0.0355N)$$

for Bessel functions, and

$$T = 4.2 + 0.172N$$

for Runge-Kutta. These were derived from the plots of the above CPU times. The cost of the additional core storage required to use the Runge-Kutta method beyond the break-even point of $N=500$ is insignificant in comparison to the increased CPU time cost required by the solution by Bessel functions.

The FORTRAN subroutine INIT for the M/M/1 queue discussed in this chapter is presented in Appendix C along with sample output used in the comparison.

The following table is a summary of benefits realized by using the Runge-Kutta method over the analytic solution involving Bessel functions. Each benefit more or less applies to every analytical technique discussed in the previous chapter.

Table 2
Summary of Benefits

<u>Runge-Kutta</u>	<u>Bessel Functions</u>
<u>analytical difficulty</u>	
No further analysis required beyond derivation of equations.	Considerable analysis to get analytic solution to equations. Analytic expression may not exist - may need approximations.
<u>programming</u>	
Easy to program for computer solution - little debugging required.	Sophisticated programming expertise required to prevent overflows, loss-of-significance, etc. Difficult to debug.
<u>numerical stability</u>	
Solution can be trusted, method behaves in a predictable way, precision can be arbitrarily chosen by user.	Numerical instabilities leave trustworthiness of solution in doubt, very little flexibility in choosing desired precision.

cost of accuracy

Cost of achieving a given level of precision is related linearly to step size over wide range of values.

Cost of achieving a given level of precision usually is related non-linearly in a power function over a narrow range of precision.

analysts' time

Time to obtain solution, given a model, is short.

Time to obtain solution, given a model, may be very long.

clarity of methods

Method and solution easily understood by others.

Method chosen and implementation difficult for others to understand.

cost of solutions

Cost of solution increases linearly with N , the number of equations solved.

Cost of solution increases as a power function of N .

size of programs

Load module of equal size occurs at approxi-

Load module size does not change with N ; greater for

mately $N=500$.

$N < 500$ than the Runge-Kutta load module.

Monte-Carlo Simulation

A simulation of the M/M/1 model was written in SIMSCRIPT-II (9). The listing and sample output are contained in Appendix A. For a sample size of 1000 replications, consisting of 10 runs of 100 replications each, the 95% confidence interval on the expected number in the system n was obtained as

$$1.66 \leq n \leq 2.32.$$

Although this is quite a large variability, it is acceptable to achieve this level in complex simulations. To make a comparison with the other two methods on the basis of computer resources, the load module size was 48,000 bytes and the total CPU time was 28 seconds. This compares favorably with Runge-Kutta and shows the efficiency of the SIMSCRIPT-II system. The CPU time, of course, would depend linearly on the number of replications. The time to program and debug the simulation was slightly less than that required for the Bessel function program.

Although not strictly a numerical technique, the simulation was included to demonstrate its relationship with the previous two methods because simulation is usually

regarded as an expensive and inefficient method to solve such problems.

CHAPTER 4

EXAMPLES OF SOLUTIONS TO OTHER QUEUEING MODELS

In this chapter two other applications of the Runge-Kutta method will be presented. The purpose is to show how simple and straightforward the preparation for solution of different problems can be when using the Runge-Kutta control program in Appendix C. The first is the solution of the preemptive-resume priority queue and the second is the solution of the state-dependent queue. These were chosen as further examples of solution by the Runge-Kutta integration because they are representative of commonly-used models.

Preemptive-Resume Queue

The preemptive-resume queueing model is a slight variation of the M/M/1. In it, the possibility of a different kind of priority arrival is allowed in the system with a different arrival rate than the normal arrivals. When a priority arrival occurs, service is suspended if service is being performed on a normal service item, and it is set aside so that service may begin on the priority arrival. Service proceeds at a different rate for the priority item than for normal items. The server continues

to serve from the priority queue until empty, at which time the normal item that had been set aside will return to service and continue where it left off.

This is an excellent model for describing how a M/M/1 model would behave if the server were subject to breakdown and repairs were made to restore it to service.

In the model that follows λ and μ are the arrival and departure rates, respectively, for the normal system and λ_p and μ_p are parameters for the priority system. The equations are written for a finite number M in the priority system and N for the normal system. Moreover the subscript i stands for the number in the priority system and j stands for the number in the normal system.

$$\dot{P}_{i,j}(t) = -(\lambda + \lambda_p + \mu_p)P_{i,j}(t) + \lambda_p P_{i-1,j}(t) + \mu_p P_{i+1,j}(t) + \lambda P_{i,j-1}(t)$$

$$\dot{P}_{0,0}(t) = -(\lambda + \lambda_p)P_{0,0}(t) + \mu_p P_{1,0}(t) + \mu P_{0,1}(t)$$

$$\dot{P}_{0,j}(t) = -(\lambda + \lambda_p + \mu)P_{0,j}(t) + \lambda P_{0,j-1}(t) + \mu P_{0,j+1}(t) + \mu_p P_{1,j}(t)$$

$$\dot{P}_{i,0}(t) = -(\lambda + \lambda_p + \mu_p)P_{i,0}(t) + \lambda_p P_{i-1,0}(t) + \mu_p P_{i+1,0}(t)$$

$$\dot{P}_{M,N}(t) = -(\lambda + \lambda_p + \mu_p)P_{M,N}(t) + \lambda_p P_{M-1,N}(t) + \lambda P_{M,N-1}(t)$$

$$\dot{P}_{0,N}(t) = -(\lambda + \lambda_p + \mu)P_{0,N}(t) + \lambda P_{0,N-1}(t) + \mu_p P_{1,N}(t)$$

$$\dot{P}_{M,0}(t) = -(\lambda + \lambda_p + \mu_p)P_{M,0}(t) + \lambda_p P_{M-1,0}(t)$$

$$\dot{P}_{M,j}(t) = -(\lambda + \lambda_p + \mu_p)P_{M,j}(t) + \lambda_p P_{M-1,j}(t) + \lambda P_{M,j-1}(t)$$

Subroutine INIT for use with the program in Appendix C and sample output follow, along with a plot of some auxiliary queue statistics in Figure 3.

PAGE 0002

18/02/31

DATE = 72103

INIT

FORTRAN IV G LEVEL 20

```

0035      3 FORMAT (1H1 'XINT 4X6HEISYS) 3X7HSTD DEV 6X4HAY Q 3X7HSTD DEV
0036      1 3X7HEISYS) 3X7HSTD DEV 5X5HAY QP 3X7HSTD DEV 3X7H8 ERROR)
0037      4 FORMAT (1H1 F10.2,8F10.5,F10.3)
      RETURN

```

```

C
C*** THE SECOND ENTRY USED TO COMPUTE THE DIFF.-DIFF. EQUATIONS
C

```

```

0038      ENTRY PRIME(T,PT,K)
0039      REAL P112,21,21

```

```

C
C*** DURING THE SOLUTION, IN THIS ENTRY POINT SECTION, THE PI-ARRAY
C*** IS USED RATHER THAN THE P-ARRAY BECAUSE WE ARE NOW DEALING
C*** WITH DIFFERENT STORAGE LOCATIONS -- IN THE WORKING STORAGE AREA.
C

```

```

0040      P11(1,1,2)=RHS(MUP*PT(2,1,1)+MUP*PT(1,2,1)-CLLP*PT(1,1,1))
0041      IF (MMO.LE.1 .OR. MMO.LE.1) GO TO 15
0042      DO 5 I=2,MMO
0043      GO 5 J=2,MMO
0044

```

```

      5 P11(J,2)=RHS(LAMP*PT(1,1,J,1)-CLLP*PT(1,J,1)+MUP*PT(1,1,J,1)
      .+LAMP*PT(1,J,1,1))
      15 IF (MMO.LI.2) GO TO 16
      DO 6 I=2,MMO
0045

```

```

      P11(1,1,2)=RHS(LAMP*PT(1,1,1,1)-CLLP*PT(1,1,1)+MUP*PT(1,1,1,1))
0046      P11(1,N,2)=RHS(LAMP*PT(1,1,N,1)-CLLP*PT(1,N,1)+MUP*PT(1,1,N,1))
0047
      .+LAMP*PT(1,1,N,1))
0048

```

```

      16 IF (MMO.LI.2) GO TO 17
      DO 7 J=2,MMO
      P11(J,2)=RHS(LAMP*PT(1,J,1,1)-CLLP*PT(1,J,1)+MUP*PT(1,J,1,1)
      .+MUP*PT(2,J,1))
0049

```

```

      7 P11(J,2)=RHS(LAMP*PT(1,J,1,1)+LAMP*PT(MMO,J,1)-CLLP*PT(M,J,1))
0050      17 P11(M,2)=RHS(LAMP*PT(MMO,N,1)-CLLP*PT(M,N,1)+LAMP*PT(M,N,1))
0051      P11(M,2)=RHS(LAMP*PT(1,MMO,1)-CLLP*PT(1,N,1)+MUP*PT(1,2,N,1))
0052      P11(1,2)=RHS(LAMP*PT(MMO,1,1)-CLLP*PT(M,1,1))
0053      RETURN
0054

```

```

C
C*** THE THIRD ENTRY POINT IS USED AFTER EACH TIME STEP WITH NEW
C*** VALUES OF THE SOLUTION. AT THIS TIME ANY OTHER VARIABLES, WHICH
C*** DEPEND ON THE P-VECTOR (EXPECTED VALUES, STANDARD DEVIATIONS,
C*** ETC.) MAY BE COMPUTED AS NECESSARY.
C

```

```

0057      ENTRY NEWVAL(T,PT)
0058      CALL CONTRLIPRINT,NPRINT,C21)
0059      ESYS=0.0
0060      VSYS=0.0
0061      AVO=G.0
0062      VJ=0.0
0063      ESYSP=G.0
0064      VSYSP=0.0

```

PAGE 0003

18/02/31

DATE = 72103

INIT

FCRTRAN IV G LEVEL 20

0065 AVQP=C.0
 0066 VQP=O.0
 0067 FJL=O.0
 0068 FJ=O.C
 0069 DJ 6 J=1,N
 0070 SUM=O.0
 0071 GO 9 I=1,M
 0072 SUM=SUM+PI(I,J,1)
 0073 ESY=ESYS+FJ*SUM
 0074 VSY=VSYSP+FJ*VQ*SUM
 0075 FJ=FJ+1.0
 0076 IF (J-EQ.1) GO TO 8
 0077 AVQ=AVQ+FJ*SUM
 0078 VQ=VQ+FJ*VQ*SUM
 0079 FJL=FJL+1.0
 0080 8 CONTINUE
 0081 FJL=O.0
 0082 FI=O.0
 0083 PSUM=O.0
 0084 DO 10 I=1,N
 0085 SUM=O.0
 0086 DO 11 J=1,N
 0087 PSUM=PSUM+PI(I,J,1)
 0088 SUM=SUM+PI(I,J,1)
 0089 ESYSP=ESYS+FI*SUM
 0090 VSYSP=VSYSP+FI*VQ*SUM
 0091 FI=FI+1.0
 0092 IF (I-EQ.1) GO TO 10
 0093 AVQ=AVQ+FI*SUM
 0094 VQ=VQ+FI*VQ*SUM
 0095 FI=FI+1.0
 0096 10 CONTINUE
 0097 VSY=SUM*(VSYSP-ESYS**2)
 0098 VQ=SUM*(VQ*VQ-PSUM**2)
 0099 VSYSP=SUM*(VSYSP-ESYS**2)
 0100 VQ=SUM*(VQ*VQ-PSUM**2)
 0101 PSUM=100.0*(PSUM-1.0)
 0102 CALL CONTROL(LINE,55,C12)
 0103 PRINT 3
 0104 12 PRINT 4,I,ESYS,VSY,AVQ,VQ,ESYSP,VSYSP,AVQP,VQP,PSUM
 0105 21 RETURN
 0106 20 CALL EXIT

C
 C*** THE FOURTH ENTRY POINT IS USED TO INDICATE IF ANOTHER
 C*** PROBLEM IS TO BE SOLVED, IN WHICH CASE EXECUTING A RETURN WILL
 C*** CAUSE THE NEXT ENTRY TO BE MADE AT INIT; OR TO TERMINATE, IN WHICH
 C*** CASE A STOP OR CALL EXIT COULD BE EXECUTED.
 C

Reproduced from
 best available copy.

FORTRAN IV G LEVEL 20
0107 ENTRY WRAPUP
0108 RETURN
0109 END

INIT

DATE = 72103

18/02/31

PAGE 0004

PREMPTIVE-RESUME MODEL WITH PRIORITY SYSTEM HAVING 2 STATES. S. OLSON APRIL 1972

LAMBDA
0.067000

LAMBDA-P
0.001670

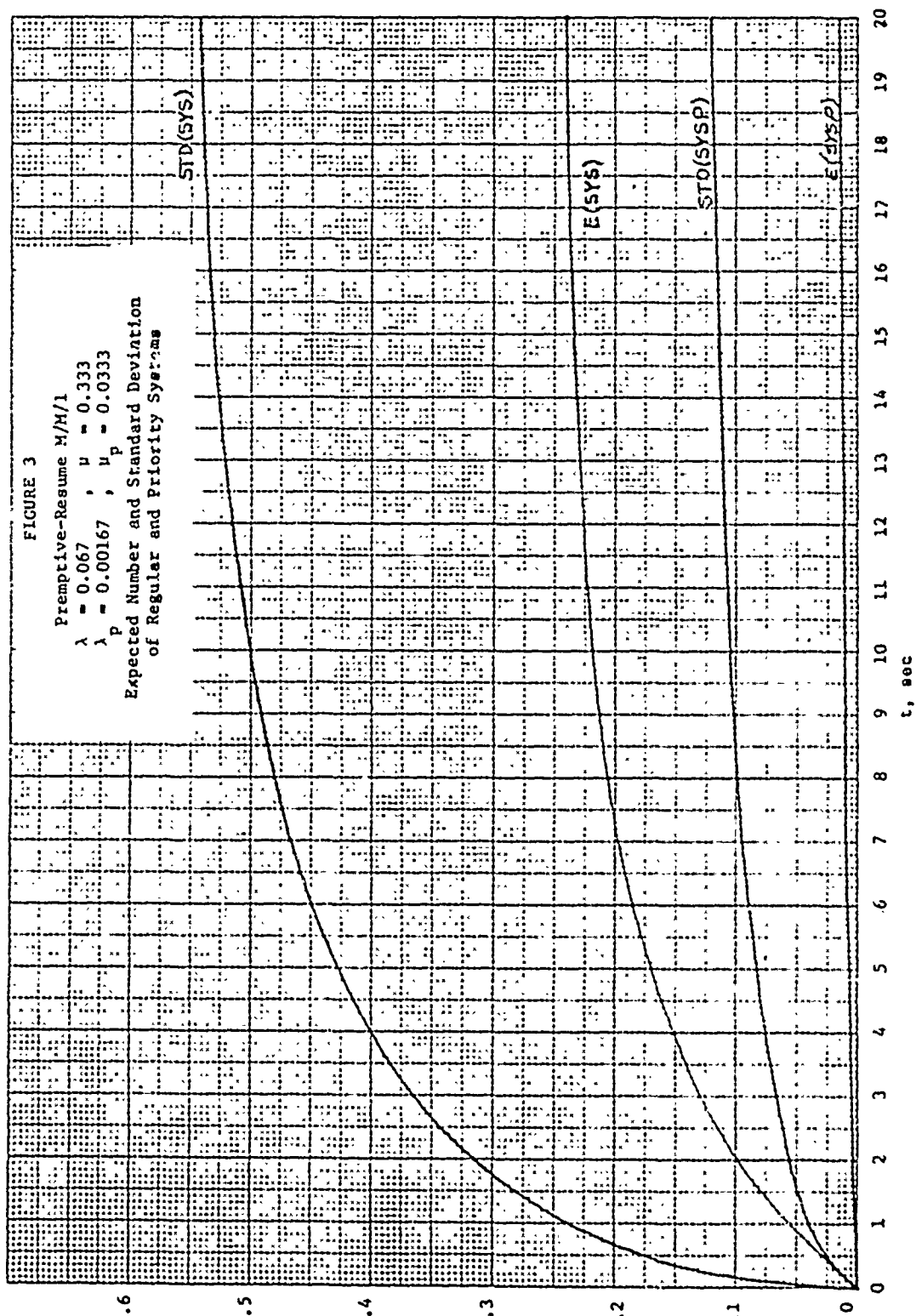
MU
0.333000

MU-P
0.033300

T	F(SYS)	STD DEV	4 th Q	STD DEV	F(SYSP)	STD DEV	AV	QP	STD DEV	Z	ERROR
0.50	0.03689	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.00	0.05118	0.117529	0.06050	0.02254	0.00083	0.02873	0.0	0.0	0.0	-0.000	-0.000
1.50	0.07098	0.273985	0.06177	0.04295	0.00163	0.04034	0.0	0.0	0.0	-0.001	-0.001
2.00	0.07992	0.27403	0.07356	0.06155	0.00241	0.04901	0.0	0.0	0.0	-0.003	-0.003
2.50	0.07932	0.11737	0.06567	0.07822	0.00316	0.05628	0.0	0.0	0.0	-0.007	-0.007
3.00	0.11274	0.05412	0.07896	0.09366	0.00387	0.06210	0.0	0.0	0.0	-0.013	-0.013
3.50	0.1325	0.1777	0.07334	0.10731	0.00455	0.06733	0.0	0.0	0.0	-0.021	-0.021
4.00	0.14272	0.1457	0.07274	0.11504	0.00529	0.07195	0.0	0.0	0.0	-0.031	-0.031
4.50	0.15357	0.08852	0.07413	0.13165	0.00582	0.07628	0.0	0.0	0.0	-0.043	-0.043
5.00	0.1647	0.172	0.07834	0.14177	0.0061	0.07918	0.0	0.0	0.0	-0.057	-0.057
5.50	0.1727	0.174	0.0809	0.1515	0.00626	0.08314	0.0	0.0	0.0	-0.074	-0.074
6.00	0.1804	0.171	0.0847	0.16181	0.00728	0.08619	0.0	0.0	0.0	-0.092	-0.092
6.50	0.1875	0.176	0.0875	0.17083	0.00798	0.08897	0.0	0.0	0.0	-0.112	-0.112
7.00	0.194	0.177	0.092	0.17921	0.00865	0.09192	0.0	0.0	0.0	-0.135	-0.135
7.50	0.2027	0.177	0.097	0.18734	0.0089	0.09365	0.0	0.0	0.0	-0.159	-0.159
8.00	0.2103	0.177	0.097	0.194	0.0090	0.09479	0.0	0.0	0.0	-0.185	-0.185
8.50	0.216	0.177	0.097	0.20272	0.00916	0.09942	0.0	0.0	0.0	-0.212	-0.212
9.00	0.22412	0.177	0.097	0.21034	0.00941	0.10191	0.0	0.0	0.0	-0.242	-0.242
9.50	0.23124	0.177	0.097	0.21764	0.00974	0.10438	0.0	0.0	0.0	-0.272	-0.272
10.00	0.23744	0.177	0.097	0.22457	0.01014	0.10654	0.0	0.0	0.0	-0.304	-0.304
10.50	0.24257	0.177	0.097	0.23115	0.01052	0.10859	0.0	0.0	0.0	-0.337	-0.337
11.00	0.2477	0.177	0.097	0.23789	0.01087	0.11052	0.0	0.0	0.0	-0.372	-0.372
11.50	0.25282	0.177	0.097	0.24457	0.01118	0.11227	0.0	0.0	0.0	-0.408	-0.408
12.00	0.25768	0.177	0.097	0.25115	0.01141	0.11382	0.0	0.0	0.0	-0.444	-0.444
12.50	0.26236	0.177	0.097	0.25765	0.01152	0.11529	0.0	0.0	0.0	-0.482	-0.482
13.00	0.26688	0.177	0.097	0.26391	0.01155	0.11643	0.0	0.0	0.0	-0.521	-0.521
13.50	0.27126	0.177	0.097	0.26980	0.01155	0.11739	0.0	0.0	0.0	-0.561	-0.561
14.00	0.27551	0.177	0.097	0.27561	0.01155	0.11827	0.0	0.0	0.0	-0.601	-0.601
14.50	0.27964	0.177	0.097	0.28139	0.01155	0.11910	0.0	0.0	0.0	-0.642	-0.642
15.00	0.28367	0.177	0.097	0.28706	0.01155	0.11988	0.0	0.0	0.0	-0.684	-0.684
15.50	0.28751	0.177	0.097	0.29269	0.01155	0.12061	0.0	0.0	0.0	-0.727	-0.727
16.00	0.29125	0.177	0.097	0.29828	0.01155	0.12129	0.0	0.0	0.0	-0.771	-0.771
16.50	0.29495	0.177	0.097	0.30384	0.01155	0.12192	0.0	0.0	0.0	-0.815	-0.815
17.00	0.29852	0.177	0.097	0.30937	0.01155	0.12257	0.0	0.0	0.0	-0.859	-0.859
17.50	0.30207	0.177	0.097	0.31487	0.01155	0.12317	0.0	0.0	0.0	-0.904	-0.904
18.00	0.30557	0.177	0.097	0.32034	0.01155	0.12373	0.0	0.0	0.0	-0.950	-0.950
18.50	0.30903	0.177	0.097	0.32577	0.01155	0.12426	0.0	0.0	0.0	-0.996	-0.996
19.00	0.31243	0.177	0.097	0.33117	0.01155	0.12477	0.0	0.0	0.0	-1.042	-1.042
19.50	0.31579	0.177	0.097	0.33654	0.01155	0.12527	0.0	0.0	0.0	-1.090	-1.090
20.00	0.31912	0.177	0.097	0.34187	0.01155	0.12577	0.0	0.0	0.0	-1.137	-1.137
20.50	0.32242	0.177	0.097	0.34716	0.01155	0.12627	0.0	0.0	0.0	-1.185	-1.185

PREMPTIVE-RESUME MODEL WITH PRIORITY SYSTEM HAVING 2 STATES. S. OLSON APRIL 1972

Y	P-VECTOR	2	0.0110	3	0.1557	4	0.0024	5	0.0300	6	0.0005	7	0.0055	8	0.0001	9	0.0009	10	0.0000
17.50	1 0.7842	2	0.0110	3	0.1557	4	0.0024	5	0.0300	6	0.0005	7	0.0055	8	0.0001	9	0.0009	10	0.0000
18.00	11 0.0001	2	0.0111	3	0.1558	4	0.0024	5	0.0301	6	0.0005	7	0.0055	8	0.0001	9	0.0010	10	0.0000
18.50	11 0.0002	2	0.0112	3	0.1558	4	0.0024	5	0.0302	6	0.0005	7	0.0056	8	0.0001	9	0.0010	10	0.0000
19.00	11 0.0002	2	0.0113	3	0.1558	4	0.0024	5	0.0303	6	0.0005	7	0.0056	8	0.0001	9	0.0010	10	0.0000
19.50	11 0.0002	2	0.0114	3	0.1557	4	0.0025	5	0.0304	6	0.0005	7	0.0057	8	0.0001	9	0.0010	10	0.0000
20.00	11 0.0002	2	0.0114	3	0.1557	4	0.0025	5	0.0304	6	0.0005	7	0.0057	8	0.0001	9	0.0010	10	0.0000



State-Dependent Queue

The model discussed here is a generalized, single-channel, state-dependent queue with s servers developed by Conway and Hillier (4). The differential equations, in terms of the independent variable $\tau = \mu t$, are

$$\dot{P}_0(\tau) = \eta_1 P_1(\tau) - \rho_0(\tau) P_0(\tau)$$

$$\dot{P}_n(\tau) = \rho_{n-1}(\tau) P_{n-1}(\tau) - [\rho_n(\tau) + \eta_n] P_n(\tau) + \eta_{n+1} P_{n+1}(\tau),$$

$$\text{for } 1 \leq n \leq N$$

$$\dot{P}_N(\tau) = \rho_{N-1}(\tau) P_{N-1}(\tau) - [\rho_N(\tau) + \eta_N] P_N(\tau),$$

where N is the maximum number of items in the system,

$$\rho_n(\tau) = \lambda_n(\tau) / \mu,$$

$$\eta_n = \mu_n / \mu = \begin{cases} n, & n < s \\ \left(\frac{n}{s}\right)^c s; & n \geq s \end{cases}$$

$$\lambda_n(\tau) = \begin{cases} \lambda(\tau) , & n < s \\ \left(\frac{s}{n+1}\right)^b \lambda(\tau) , & n \geq s. \end{cases}$$

This model becomes the standard M/M/1 queue when the parameters b and c are zero and $s=1$. This is a useful model for working with an existing system for which empirical data are available. The time dependence of arrivals and departures can be handled as well as a wide variety of system dependence through the choice of b , c and s .

Here $\rho = \lambda/\mu$ is called the loading factor. A steady-state solution can not be obtained when $\rho > 1$. However, it is often required to study systems for which $\rho \geq 1$ during part of the time of interest. Figure 4 shows how a single channel dual server queue model behaves with $\rho = 1.5$ from $\tau = 0$ until $\tau = 8$ with the system initially empty.

Subroutine INIT for this model and sample output are included next. Note that T in the output refers to τ above.

PAGE 0001

18/09/54

DATE = 72103

INIT

FORTRAN IV G LEVEL 20

C001 SURROUTINE INIT

C
C*** STATIC-DEPENDENT MODEL.
C*** INITIALIZATION SECTION INCLUDES COMPUTATION OF DIMENSIONLESS
C*** PARAMETERS WHICH ARE ONLY FUNCTIONS OF N.

C
C*** IN THIS MODEL, RHO WILL BE SET AT 1.5 FOR OCT<8. THEN RHO WILL
C*** BE SET TO 2.5 TO EXPLORE HOW THE SYSTEM RECOVERES FROM OVERLOAD.
C*** 1 IS TAU IN THE EQUATIONS, THEREFORE MU IS ACCOUNTED FOR IN T.
C*** S=2 SAVES WILL BE USED.

C002 COMMON TITLE(20),TO,TMAX,DT,NPRINT,NEQ,P(21,2),M(63)
C003 REAL PUNT(21),ETAN(21),LAMBDA,PV
C004 1C READ (5,1,END=11) TITLE,TO,TMAX,DT,NPRINT,NEQ,RH01,RH02,B,C,S
C005 1 F0PMAT (20A4/3F10.0,5X215/5F10.0)
C006 N=NCO-1
C007 P(1,1)=1.0
C008 DO 2 I=2,NEQ
C009 2 P(1,1)=0.0

C
C*** COMPUTE FUNCTIONS ONLY OF N.

C010 ETAN(1)=1.0
C011 RHO(1)=1.0
C012 FS=S
C013 FN=1.0
C014 DO 3 I=2,MFO
C015 F2=F1+1.0
C016 IF (FH.GE.FS) GO TO 4
C017 RHO(1)=1.0
C018 ETAN(1)=FN
C019 GO TO 3
C020 4 ETAN(1)=(F1/FS)*C*S
C021 RHO(1)=(F1/FS)*B
C022 3 FN=F2
C023 WRITE (6,5) TITLE,TO,TMAX,DT,NPRINT,NEQ,RH01,RH02,B,C,S
C024 5 F0PMAT (1H120A4/1H013X2H10 11X4HTMAX 13X2HDT 10X5HNPRINT 12X3HNEQ/
C . /1H 5F15.4)
C LINE=55
C IPRINT=0
C RETURN

C
C025
C026
C027
C028
C029
C030
C031
C032
C033
C034
C035
C036
C037
C038
C039
C040
C041
C042
C043
C044
C045
C046
C047
C048
C049
C050
C051
C052
C053
C054
C055
C056
C057
C058
C059
C060
C061
C062
C063
C064
C065
C066
C067
C068
C069
C070
C071
C072
C073
C074
C075
C076
C077
C078
C079
C080
C081
C082
C083
C084
C085
C086
C087
C088
C089
C090
C091
C092
C093
C094
C095
C096
C097
C098
C099
C100
C101
C102
C103
C104
C105
C106
C107
C108
C109
C110
C111
C112
C113
C114
C115
C116
C117
C118
C119
C120
C121
C122
C123
C124
C125
C126
C127
C128
C129
C130
C131
C132
C133
C134
C135
C136
C137
C138
C139
C140
C141
C142
C143
C144
C145
C146
C147
C148
C149
C150
C151
C152
C153
C154
C155
C156
C157
C158
C159
C160
C161
C162
C163
C164
C165
C166
C167
C168
C169
C170
C171
C172
C173
C174
C175
C176
C177
C178
C179
C180
C181
C182
C183
C184
C185
C186
C187
C188
C189
C190
C191
C192
C193
C194
C195
C196
C197
C198
C199
C200
C201
C202
C203
C204
C205
C206
C207
C208
C209
C210
C211
C212
C213
C214
C215
C216
C217
C218
C219
C220
C221
C222
C223
C224
C225
C226
C227
C228
C229
C230
C231
C232
C233
C234
C235
C236
C237
C238
C239
C240
C241
C242
C243
C244
C245
C246
C247
C248
C249
C250
C251
C252
C253
C254
C255
C256
C257
C258
C259
C260
C261
C262
C263
C264
C265
C266
C267
C268
C269
C270
C271
C272
C273
C274
C275
C276
C277
C278
C279
C280
C281
C282
C283
C284
C285
C286
C287
C288
C289
C290
C291
C292
C293
C294
C295
C296
C297
C298
C299
C300
C301
C302
C303
C304
C305
C306
C307
C308
C309
C310
C311
C312
C313
C314
C315
C316
C317
C318
C319
C320
C321
C322
C323
C324
C325
C326
C327
C328
C329
C330
C331
C332
C333
C334
C335
C336
C337
C338
C339
C340
C341
C342
C343
C344
C345
C346
C347
C348
C349
C350
C351
C352
C353
C354
C355
C356
C357
C358
C359
C360
C361
C362
C363
C364
C365
C366
C367
C368
C369
C370
C371
C372
C373
C374
C375
C376
C377
C378
C379
C380
C381
C382
C383
C384
C385
C386
C387
C388
C389
C390
C391
C392
C393
C394
C395
C396
C397
C398
C399
C400
C401
C402
C403
C404
C405
C406
C407
C408
C409
C410
C411
C412
C413
C414
C415
C416
C417
C418
C419
C420
C421
C422
C423
C424
C425
C426
C427
C428
C429
C430
C431
C432
C433
C434
C435
C436
C437
C438
C439
C440
C441
C442
C443
C444
C445
C446
C447
C448
C449
C450
C451
C452
C453
C454
C455
C456
C457
C458
C459
C460
C461
C462
C463
C464
C465
C466
C467
C468
C469
C470
C471
C472
C473
C474
C475
C476
C477
C478
C479
C480
C481
C482
C483
C484
C485
C486
C487
C488
C489
C490
C491
C492
C493
C494
C495
C496
C497
C498
C499
C500
C501
C502
C503
C504
C505
C506
C507
C508
C509
C510
C511
C512
C513
C514
C515
C516
C517
C518
C519
C520
C521
C522
C523
C524
C525
C526
C527
C528
C529
C530
C531
C532
C533
C534
C535
C536
C537
C538
C539
C540
C541
C542
C543
C544
C545
C546
C547
C548
C549
C550
C551
C552
C553
C554
C555
C556
C557
C558
C559
C560
C561
C562
C563
C564
C565
C566
C567
C568
C569
C570
C571
C572
C573
C574
C575
C576
C577
C578
C579
C580
C581
C582
C583
C584
C585
C586
C587
C588
C589
C590
C591
C592
C593
C594
C595
C596
C597
C598
C599
C600
C601
C602
C603
C604
C605
C606
C607
C608
C609
C610
C611
C612
C613
C614
C615
C616
C617
C618
C619
C620
C621
C622
C623
C624
C625
C626
C627
C628
C629
C630
C631
C632
C633
C634
C635
C636
C637
C638
C639
C640
C641
C642
C643
C644
C645
C646
C647
C648
C649
C650
C651
C652
C653
C654
C655
C656
C657
C658
C659
C660
C661
C662
C663
C664
C665
C666
C667
C668
C669
C670
C671
C672
C673
C674
C675
C676
C677
C678
C679
C680
C681
C682
C683
C684
C685
C686
C687
C688
C689
C690
C691
C692
C693
C694
C695
C696
C697
C698
C699
C700
C701
C702
C703
C704
C705
C706
C707
C708
C709
C710
C711
C712
C713
C714
C715
C716
C717
C718
C719
C720
C721
C722
C723
C724
C725
C726
C727
C728
C729
C730
C731
C732
C733
C734
C735
C736
C737
C738
C739
C740
C741
C742
C743
C744
C745
C746
C747
C748
C749
C750
C751
C752
C753
C754
C755
C756
C757
C758
C759
C760
C761
C762
C763
C764
C765
C766
C767
C768
C769
C770
C771
C772
C773
C774
C775
C776
C777
C778
C779
C780
C781
C782
C783
C784
C785
C786
C787
C788
C789
C790
C791
C792
C793
C794
C795
C796
C797
C798
C799
C800
C801
C802
C803
C804
C805
C806
C807
C808
C809
C810
C811
C812
C813
C814
C815
C816
C817
C818
C819
C820
C821
C822
C823
C824
C825
C826
C827
C828
C829
C830
C831
C832
C833
C834
C835
C836
C837
C838
C839
C840
C841
C842
C843
C844
C845
C846
C847
C848
C849
C850
C851
C852
C853
C854
C855
C856
C857
C858
C859
C860
C861
C862
C863
C864
C865
C866
C867
C868
C869
C870
C871
C872
C873
C874
C875
C876
C877
C878
C879
C880
C881
C882
C883
C884
C885
C886
C887
C888
C889
C890
C891
C892
C893
C894
C895
C896
C897
C898
C899
C900
C901
C902
C903
C904
C905
C906
C907
C908
C909
C910
C911
C912
C913
C914
C915
C916
C917
C918
C919
C920
C921
C922
C923
C924
C925
C926
C927
C928
C929
C930
C931
C932
C933
C934
C935
C936
C937
C938
C939
C940
C941
C942
C943
C944
C945
C946
C947
C948
C949
C950
C951
C952
C953
C954
C955
C956
C957
C958
C959
C960
C961
C962
C963
C964
C965
C966
C967
C968
C969
C970
C971
C972
C973
C974
C975
C976
C977
C978
C979
C980
C981
C982
C983
C984
C985
C986
C987
C988
C989
C990
C991
C992
C993
C994
C995
C996
C997
C998
C999
C1000

Reproduced from
best available copy.

PAGE 0002

Reproduced from
best available copy.

```

FORTRAN IV G LEVEL 20          INIT          DATE = 72103          18/09/54

0033      DO 7 I=2,N
0034      7 PV(I,2)=RHS(ETAN(I+1)*PV(I+1,1)-(ETAN(I)*RHON(I)*RHOM)*PV(I,1)
      *RHOM(I-1)*RHOM*PV(I-1,1))
0035      PV(NEQ,2)=RHS(RHON(NEQ)*RHOM*PV(NEQ,1)-(ETAN(N)*RHON(N)*RHOM)
      *PV(4,1))
0036      RETURN

C
0037      ENTRY NEWVAL(T,PV)
0038      8 CALL CUMTAL(IPRNT,NPRNT,E9)
0039      8 KPO1=PV(1,1)
0040      ESYS=Q-J
0041      STDSYS=Q-C
0042      FJ=1-C
0043      DO 12 J=2,NEQ
0044      8 KPO2=EXXOR*PV(J,1)
0045      ESYS=ESYS+FJ*PV(J,1)
0046      STDSYS=STDSYS+FJ*PV(J,1)
0047      FJ=FJ*1-C
0048      EXXOR=10C-C*(1-EXXOR-1-C)
0049      STDSYS=SQRT(ABS(STDSYS-ESYS**2))
0050      CALL CUMTAL(LINE,55,613)
0051      WRITE (6,14) TITLE
0052      14 FORMAT (1H120A4/1H9 14X1HT 11X4HESYS 9X6HSTDSYS 3X12HPERCENT ERR.)
0053      13 WRITE (6,15) T-ESYS,STDSYS,ERROR
0054      15 FORMAT (1H F15.2,3F15.6)
0055      9 RETURN

C
0056      ENTRY WRAPUP
0057      RETURN
0058      11 CALL EXIT
0059      END

```

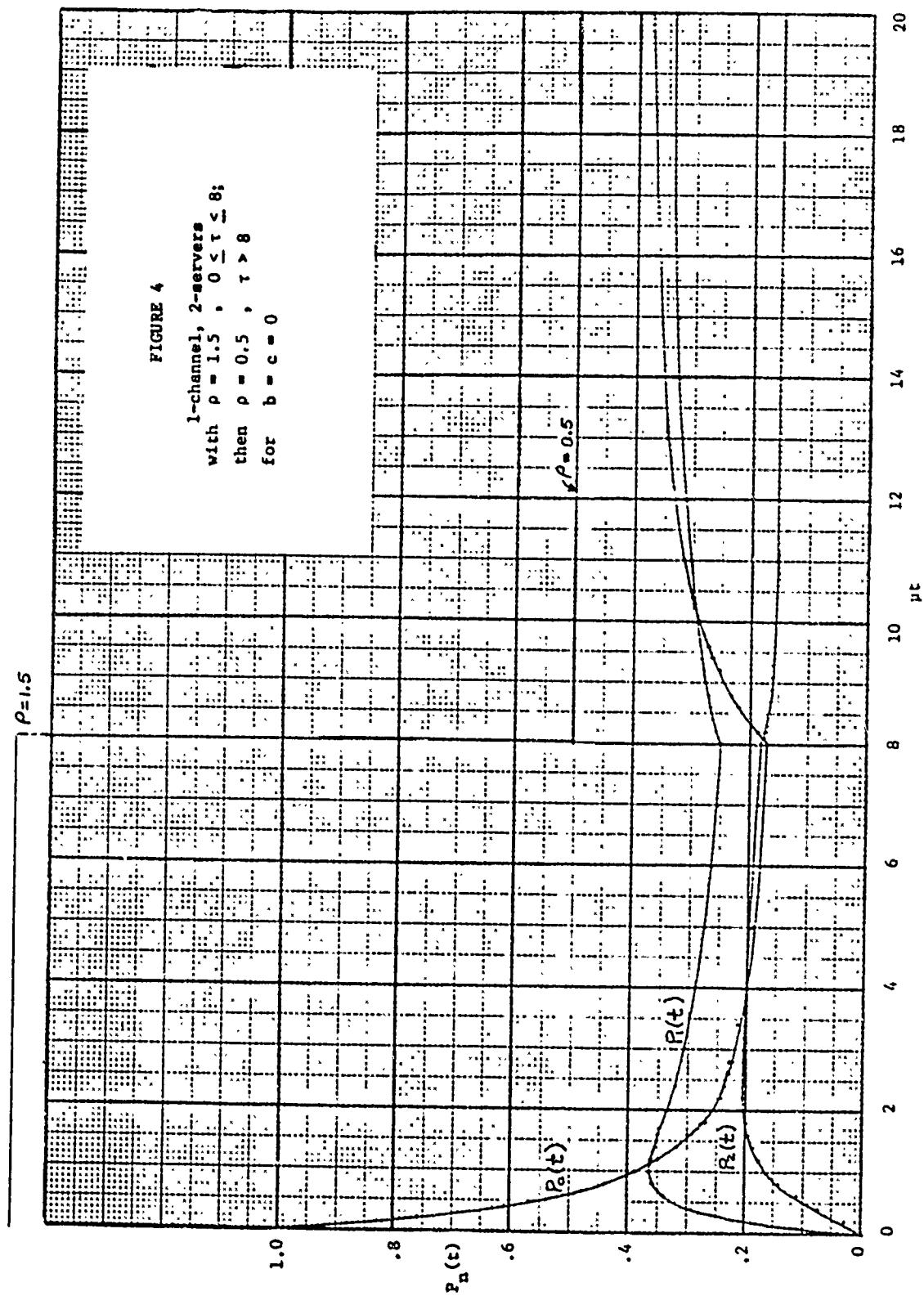
STATE-DEPENDENT QUEUE		S. OLSON		56-299	A. RIL, 1972	
TO	YMAX	DT	NPRINT	NEQ		
0.0	20.0000	0.1000	5	21		
RND(Y<8)	RND(T>8)	B	C	S		
1.5000	0.5000	0.0	0.0	2.0000		

Reproduced from
best available copy.

STATE-DEPENDENT QUEUE				S. OLSON		56-299		APRIL, 1972	
T	FSYS	STDYS	PERCENT ERR.						
0.50	0.594384	0.778937	-0.000324						
1.00	0.979432	1.033791	-0.000672						
1.50	1.252798	1.217229	-0.000125						
2.00	1.461122	1.368942	-0.000185						
2.50	1.628193	1.478303	-0.000250						
3.00	1.767780	1.611950	-0.000322						
3.50	1.885562	1.713208	-0.000393						
4.00	1.984569	1.804424	-0.000501						
4.50	2.075423	1.887289	-0.000602						
5.00	2.161451	1.963072	-0.000729						
5.50	2.233365	2.032743	-0.001121						
6.00	2.299421	2.097050	-0.001657						
6.50	2.359611	2.156566	-0.002581						
7.00	2.414698	2.211755	-0.003929						
7.50	2.465294	2.263901	-0.006092						
8.00	2.511902	2.310629	-0.008976						
8.50	2.554757	2.352742	-0.010633						
9.00	2.593101	2.390446	-0.010502						
9.50	2.627924	2.424032	-0.009739						
10.00	2.658559	2.453481	-0.009125						
10.50	2.684817	2.478881	-0.008947						
11.00	2.706632	2.500120	-0.009120						
11.50	2.723645	2.517668	-0.009412						
12.00	2.736336	2.530827	-0.009650						
12.50	2.744719	2.539774	-0.009787						
13.00	2.749113	2.544917	-0.009817						
13.50	2.749496	2.547609	-0.009823						
14.00	2.746371	2.548082	-0.009882						
14.50	2.740612	2.546402	-0.009948						
15.00	2.732900	2.542794	-0.010078						
15.50	2.723009	2.537007	-0.010031						
16.00	2.710739	2.528749	-0.010027						
16.50	2.695879	2.517420	-0.010145						
17.00	2.679025	2.502634	-0.010175						
17.50	2.660448	2.486714	-0.010210						
18.00	2.640872	2.468101	-0.010258						
18.50	2.620074	2.447331	-0.010300						
19.00	2.598001	2.424025	-0.010353						
19.50	2.574903	2.398188	-0.010377						
20.00	2.550923	2.369188	-0.010377						

STATE-DEPENDENT QUEUE S. NOLSON 56-299 APRIL, 1972

T	P-VECTOR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000
---	----------	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------



CHAPTER 5

SUMMARY

Methods for transient solutions of queueing problems have been explored and evaluated for their usefulness in numerical work. The so-called closed-form analytic solutions are the least-preferred forms for numerical work based on difficulty of analysis, computer programming and cost. Monte-Carlo simulation is easier and cheaper to apply than these analytic solutions but should not be used when the queueing model can be expressed in the form of differential-difference equations. In this case, the use of Runge-Kutta integration is preferred.

The main control program in Appendix C has been shown to be easy to use, capable of high precision, low in cost and useful in a wide range of queueing models. It is offered as a much-needed tool to researchers and practitioners in queueing theory with the hope that the use of transient analyses in queueing theory will be more easily attainable. To this extent, existing results will be made more meaningful (2).

Two areas of further research have become apparent during the preparation of this thesis. One concerns the incorporation of the Runge-Kutta queueing problem solver

described in Appendix C in continuous optimization applications. The other concerns numerical analysis on the Runge-Kutta algorithm as employed here.

The ability to obtain fast and inexpensive transient solutions to queueing problems suggests the application of variational methods to compute optimal functional solutions rather than simple scalar solutions. An example might be a problem of finding the optimal service rate of a system subject to a given fluctuating rate of arrivals and to constraints on the maximum number in the system. The optimality conditions might include a cost function relating how cost varies as a function of service rate or capacity, number in the system and the loss of arrivals when the system is full. The optimizing equation, representing the integral of the variational of the optimality conditions would then have the solution of the queueing model imbedded within its solution. A new main program to solve the queueing problem as a part of the optimization solution would be required in addition to what is presented in this thesis. Numerous optimization algorithms are available for ready incorporation in such a program.

Additional work in the numerical analysis of the Runge-Kutta algorithm was suggested in Chapter two. The Runge-Kutta algorithm can be derived in many forms depending upon the desired error in a single step and assumptions regarding the behavior of the equations being integrated. The form

employed in this thesis is one commonly-used on a great variety of problems. It can be made to work in most difficult cases by decreasing the step size until stability is eventually achieved. Generally, when this occurs, the analyst realizes the algorithm may not be appropriate and seeks another more suited to the problem.

On the other hand, it has been discovered that the Runge-Kutta equations will solve the queueing models in this thesis accurately with very large step sizes. This would seem to indicate that the method is more accurate than it need be. The differential equations in queueing are a specialized type of problem. They are usually first-order, highly-coupled and yet stable under drastically-changing conditions of the rate parameters. Future work should attempt to rederive the Runge-Kutta formula for the queueing equations. It may be possible to obtain a more economical form utilizing the characteristics of queueing models and with fewer evaluations of the differential equations at each step.

LIST OF REFERENCES

1. Beckett, R. and Hurt, J., Numerical Calculations and Algorithms, McGraw-Hill Book Company, Inc., New York, 1967.
2. Bhat, U.N., "Sixty years of queueing theory", *Management Science*, Vol. 15, No. 6, p.B280-94, 1969.
3. Clark, A.B., "The time-dependent waiting line problem", *University of Michigan Engineering Research Institute Report No. M720-1 R 39*, 1953.
4. Conway, R.W. and Hillier, F.S., "A multiple server queueing model with state dependent service rate", *Journal of Industrial Engineering*, Vol. XV, No. 3, 1964.
5. Gue, R.L. and Thomas, M.E., Mathematical Methods in Operations Research, The Macmillan Company, New York, 1968.
6. "IBM Operating System 360: FORTRAN IV Language", *IBM Systems Reference Library*, C28-6515, 1968.
7. "IBM Operating System 360: Scientific Subroutine Package", *IBM Systems Reference Library*, H20-0205, 1966.
8. Kaspar, H., "Methods for a transient solution to a queueing problem", *TRW Systems Group*, Redondo Beach, Cal., 1971.
9. Kiviat, P.J., Villanueva, R. and Markowitz, H., The SIMSCRIPT-II Programming Language, Prentice-Hall, Englewood Cliffs, N.J., 1968.
10. Leese, E.L. and Boyd, D.W., "Numerical methods of determining the transient behavior of queues with variable arrival times", *Canadian Operational Research Journal*, Vol. 4, No. 1, 1966.

11. Neuts, M.F., "The single server queue in discrete time - numerical analysis I", Mimeograph Series No. 270, Dept. of Statistics, Purdue University, Nov., 1971.
12. Newell, G.F., "Queues with time-dependent arrival rates: I-The transition thru saturation", Journal of Applied Probability, Vol. 5, p.436, 1968.
13. _____, "Queues with time-dependent arrival rates: II-The maximum queue and the return to equilibrium", Journal of Applied Probability, Vol. 5, p.579, 1968.
14. Nielsen, K.L., College Outline Series-Differential Equations, Barnes and Noble, Inc., New York, 1962.
15. Saaty, T.L., "Resume of useful formulas in queueing theory", Operations Research, Vol. 5, p.161, 1957.
16. Schlenker, G.J., "Numerical methods in renewal theory", Master's Thesis, Dept. of Industrial and Management Engineering, University of Iowa, Feb , 1968.

GENERAL REFERENCES

1. Ackoff, R.L., ed., Progress in Operations Research, Volume I, John Wiley and Sons, Inc., New York, 1961.
2. Arora, K.L., "Time-dependent solution of the two-server queue fed by general arrival and exponential service time distribution", Operations Research, Vol. 10, No. 3, 1962.
3. Bagchi, T.P. and Chakrabarti, S.K., "Bayesian approach to the design of a queueing system", Canadian Journal of Operational Research and Information Processing, Vol. 10, No. 1, 1972.
4. Bailey, N.T.J., The Elements of Stochastic Processes with Applications to the Natural Sciences, John Wiley and Sons, Inc., New York, 1964.
5. Finch, P.D., "On the transient behavior of a simple queue", Journal of the Royal Statistical Society, B22, p.277, 1960.
6. Galliher, H.P. and Wheeler, R.D., "Non-stationary queueing probabilities for landing congestion of aircraft", Operations Research, Vol. 6, p.264, 1958.
7. Gnedenko, B.V. and Kovalenko, I.N., Introduction to Queueing Theory, Weiser Military Limited, Jerusalem, Israel, 1968.
8. Greenberg, I., "Some fundamental results in the theory of queues", Journal of Applied Probability, Vol. 6, p.29, 1969.
9. Heyne, J.B. and Brotman, D., "On pulse-type arrivals to a queue", Operations Research, Vol. 8, No. 3, 1960.
10. Hillier, F.S. and Lieberman, G.J., Introduction to Operations Research, Holden-Day, Inc., San Francisco, Cal., 1968.
11. Jaiswal, N.K., Priority Queues, Academic Press, New York, 1968.

12. Luchak, G., "The solution of the single-channel queueing equations characterized by a time-dependent arrival rate and a general class of holding times", *Operations Research*, Vol. 4, p.711, 1956.
13. _____, "The distribution of the time required to reduce to some preassigned level a single-channel queue characterized by a time-dependent Poisson-distributed arrival rate and a general class of holding times", *Operations Research*, Vol. 5, p.205, 1957.
14. Morse, P.M., Queues, Inventories and Maintenance, John Wiley and Sons, Inc., New York, 1958.
15. Saaty, T.L., "Time-dependent solution of the many-server Poisson queue", *Operations Research*, Vol. 8, No. 6, 1960.
16. _____, Elements of Queueing Theory, McGraw-Hill Book Company, Inc., New York, 1961.
17. Takacs, L., "Transient behavior of a single-channel, single-server queueing process with recurrent input and exponentially distributed service times", *Operations Research*, Vol. 8, p.321, 1960.
18. _____, Introduction to the Theory of Queues, Oxford University Press, New York, 1962.

APPENDIX A
LISTING OF MONTE-CARLO SIMULATION PROGRAM

The computer program listed here is a simulation of a M/M/1 queue with the same rate parameters and maximum number of items in the system N as used in the comparison in Chapter three. The operation of the simulation is straightforward and largely self-explanatory due to the uniqueness of the SIMSCRIPT-II language. The elements of the simulation include a QUEUE which is FIFO by default, ARRIVAL events which cause ITEMS to be created and SERVICE events. When an ARRIVAL event occurs, an ITEM is created, its time of arrival is saved and it is filed in the QUEUE. The time between arrivals is an exponentially-distributed random variable computed by the EXPONENTIAL.F function. ARRIVALS reschedule themselves. A SERVICE event is triggered whenever an ITEM joins the empty QUEUE and/or when an ITEM completes SERVICE and the QUEUE is not empty. The SERVICE completion event time is also scheduled using exponentially-distributed random variables.

At pre-specified times during the time the simulation is allowed to run, events named LOOK occur. This event records the instantaneous number in the system in a histogram named P. The simulation also computes average

waiting time in the system and percent utilization of the server.

The simulation is replicated 1000 times while the histogram is accumulated. The histogram cells are then divided by 1000 to give estimates of the probability of the number in the system at a particular time and printed. The sample output shows this for times of 5, 10, 15 and 20. These times correspond to times printed in the other two methods discussed in Chapter three. The results are in fair agreement with the Runge-Kutta solution in Chapter three.

LINE CACI SIMSCRIPT II.5 VERSION 5 04/12/72 PAGE 1
 1 PREAMBLE.....
 2
 3 NORMALLY, MODE IS REAL AND DIMENSION IS 0
 4
 5 PERMANENT ENTITIES...
 6 THE SYSTEM OWNS A QUEUE
 7 EVERY SNAP HAS A T.SNAP, AN AVSYS AND A STOSYS.
 8
 9 TEMPORARY ENTITIES...
 10 EVERY ITEM HAS A T.ARRVL AND MAY BELONG TO SOME QUEUE.
 11
 12 EVENT NOTICES INCLUDE ARRIVAL, SERVICE AND LOOK.
 13 PRIORITY ORDER IS ARRIVAL, SERVICE AND LOOK
 14
 15 DEFINE BUSY,SW,HMAX,HREPS,REP,N,NITEM,SITEM,DEBUG,TOTAL
 16 AS INTEGER VARIABLES
 17 DEFINE P AS A 2-DIMENSIONAL ARRAY.
 18 DEFINE ARR-RATE,SERV-RATE,MTBA,MTBS,TEND,T.BUSY,T.WAIT,T.START,
 19 ISAVE,ICHANGE AS VARIABLES
 20
 21 DEFINE LAZY TO MEAN DEFINE % AS AN INTEGER VARIABLE.
 22 END...

Reproduced from
best available copy.

```

LINE  CACI SIMSCRIPT II.5  VERSION 5                04/12/72  PAGE  2

1  MAIN.....
2  LAZY
3
4  *NEXTCASE*
5  IF DATA IS ENDED STOP
6  OTHERWISE
7
8  READ APR,PAIF, SERV.PATE,NMAX,N, SNAP,NREPS AND DEBUG
9  LET N=NMAX+1
10 CREATE EVERY SNAP
11 RESERVE P AS N BY N.SNAP
12 FOR EACH SNAP, READ I,SNAP(SNAP)
13 LET I=I-SNAP(I,SNAP)
14 FOR EACH SNAP, DO
15   LET AVSYS(SNAP)=0.0
16   LET STDSYS(SNAP)=0.0
17   FOR I=1 TO N, LET P(I,SNAP)=0.0
18   LOOP
19   LET MICA=1.0/ARR.RATE
20   LET MIRS=1.0/SERV.RATE
21   LET I.BUSY=0.0
22   LET T.WAIT=0.0
23   LET TOTAL=0
24
25   FOR REP=1 TO NREPS, DO
26     LET TIME.V=0.0
27     LET SNAP=1
28     LET SITEM=0
29     LET TSAVE=C.0
30     LET TCHANGE=C.0
31     LET NITEM=N
32     SCHEDULE A, ARRIVAL NEXT
33     SCHEDULE A LOOK AT T.SNAP(SNAP)
34     LET BUSY.SN=N
35
36     START SIMULATION
37
38   *AGAIN* IF QUEUE IS NOT EMPTY,
39     REMOVE THE FIRST ITEM FROM QUEUE
40     SUBTRACT 1 FROM NITEM
41     DESTROY THIS ITEM
42     GO AGAIN
43   OTHERWISE
44     SUBTRACT BUSY.SN FROM NITEM
45     ADD NITEM TO TOTAL
46   LOOP
47   START NEW PAGE
48   PRINT 4 LINES WITH ARR.RATE, SERV.RATE,NMAX,NREPS,
49     T.BUSY/TIME.V/NREPS*100.0 AND T.WAIT/TOTAL AS FOLLOWS
50
51 SIMULATION OF SINGLE-CHANNEL, SINGLE-SERVER QUEUE  S.OLSON,56-299,APRIL,1972
52
53 ARR. RATE SER. RATE QUE. LIMIT REPLICATIONS 3 SERV. UTILIZATION  AVG. WAIT
54 .....  .....  .....  .....  .....  .....

```

```

50 LINE CACI SIMSCRIPT II.5 VERSION 5 04/12/72 PAGE 3
51
52 FOR EACH SNAP, DO
53 LET AVG=AVSYS(SNAP)/NREPS
54 LET SID=SQRT((ABS.FSTDYS(SNAP)-NREPS*AVG)/(NREPS-1))
55 START NEW PAGE
56 PRINT 7 LINES WITH I,SNAP(SNAP), AVG AND STD AS FOLLOWS
57     AVG, IN SYS=,,,,, STD.DEV=,,,,,
58     FOR I=1 TO N, PRINT I LINE WITH I-1,PII,SNAP,NREPS AS FOLLOWS
59     P(=)=,.,,.,,.,,
60 LOOP
61 RELEASE P,I,SNAP,AVG,S AND STD,SYS
62 GO TO NEXCASE
63 END,.,,.,,.,,

```

```

LINE  CACT SIMSCRIPT 11-5  VERSION 5                04/12/72  PAGE  4

1  UPON ARRIVAL SAVING THE EVENT NOTICE...
2
3  LET INXT=TIME.V*EXPONENTIAL.F(IHTBA,1)
4  IF INXT IS GREATER THAN TEND
5    DESTROY THIS ARRIVAL
6    GO JOIN*
7  OTHERWISE
8    RESCHEDULE THIS ARRIVAL AT INXT
9  *JOIN*
10 IF N.QUEUE EQUALS NMAX RETURN
11 OTHERWISE
12   CREATE AN ITEM
13   LET T.ARRVL(IITEM)=TIME.V
14   IF BUSY-SW IS ZERO
15     SCHEDULE A SERVICE NEXT
16     REGAPDLESS
17     FILE THE ITEM IN QUEUE
18     ADD 1 TO NITEM
19     LET NSYS=N.QUEUE+BUSY-SW
20     ADD NSYS*TIME.V-TCCHANGE) TO ISAVE
21     LET TCHANGE=TIME.V
22
23   IF DEBUG IS ZERO RETURN
24   OTHERWISE
25     PRINT 1 LINE WITH TIME.V, NITEM,N-QUEUE THUS
AT TIME ***.***, ITEM ** JOINS QUEUE; LENGTH*** -----ARRIVAL
26     RETURN
27   END.....

```

```

LINE  CACI SIMSCRIPT II.5  VERSION 5
1  UPON SERVICE SAVING THE EVENT NOTICE...
2
3  LET NSYS=N.QUEUE+BUSY.SW
4  ADD NSYS*(TIME-V-TCCHANGE) TO TSAVE
5  LET TCHNGI=TIME-V
6  LET ELAPSED=TIME-V-T.START
7  ADD ELAPSED TO T.BUSY
8  ADD ELAPSED TO T.WAIT
9
10 IF QUEUE IS EMPTY,
11   DESTROY THIS SERVICE
12   LET BUSY.SW=0
13   RETURN
14 OTHERWISE
15
16 IF DEBUG IS ZERO GO NORMALLY
17 OTHERWISE
18   ADD 1 TO SITEM
19   PRINT 1 LINE WITH TIME,V,SITEM,N.QUEUE AS FOLLOWS
20   AT TIME ***.***, ITEM *** BEGINS SERVICE, LENGTH=*** -----SERVICE
21   NORMALLY.
22
23 REMOVE THE FIRST ITEM FROM QUEUE
24 ADD TIME-V-T.ARRVL(ITEM) TO T.WAIT
25 DESTROY THIS ITEM
26 LET TNEXT=TIME-V*EXPONENTIAL.F(MTBS,2)
27 LET BUSY.SW=1
28 LET T.START=TIME-V
29 IF TNEXT IS GREATER THAN TEND
30   DESTROY THIS SERVICE
31   RETURN
32 OTHERWISE
33   RESCHEDULE THIS SERVICE AT TNEXT
34   RETURN
35 END.....

```

04/12/72 PAGE 5

```

LINE  CACT SIMSCRIPT II-5  VERSION 5
1  UPON LOOK SAVING THE EVENT NOTICE...
2
3  LAZY
4  IF SNAP EQUALS N-SNAP
5  DESTROY THIS LOOK
6  GO TO GO-TO-AD
7  OTHERWISE
8  RESCHEDULE THIS LOOK AT T-SNAP(SNAP+1)
9  *RECORD*
10 LET T-N-OUTQUE=BUSY-SW
11 ADD 1 TO AVSYST(SNAP)
12 ADD 1+1 TO SYSTSYS(SNAP)
13 ADD 1-5 TO P(1+1-SNAP)
14 ADD 1 TO SNAP
15
16 IF DEFING IS ZERO RETURN
17 OTHERWISE
18 PRINT 1 LINE WITH TIME-V-SNAP-11 AS FOLLOWS
19 AT TIME ****, SNAPSHOT *** WITH *** IN SYSTEM -----LOOK
20 RETURN
21 END.....

```

SIMULATION OF SINGLE-CHANNEL, SINGLE-SERVER QUEUE S. OLSON, 56-299, APRIL, 1972

ARR. RATE	SER. RATE	QUE. LIMIT	REPLICATIONS	% SERV. UTILIZATION	AVG. WAIT
.450	.500	20	1000	13.1473	2.501

TIME= 5.0000 AV. NO. IN SYS.= 1.565 STD. DEV.= 1.541

P(0)= .301200
P(1)= .264000
P(2)= .196000
P(3)= .127000
P(4)= .063000
P(5)= .032000
P(6)= .010000
P(7)= .004000
P(8)= .001000
P(9)= .001000
P(10)=0.
P(11)= .001000
P(12)=0.
P(13)=0.
P(14)=0.
P(15)=0.
P(16)=0.
P(17)=0.
P(18)=0.
P(19)=0.
P(20)=0.

TIME= 10.0000 AV.NO. IN SYS.= 1.989 STD.DEV.= 1.912

P(0)= .269000
P(1)= .215000
P(2)= .189000
P(3)= .128000
P(4)= .087000
P(5)= .054000
P(6)= .029000
P(7)= .020000
P(8)= .004000
P(9)= .003000
P(10)= .001000
P(11)= .001000
P(12)=0.
P(13)=C.
P(14)=G.
P(15)=0.
P(16)=0.
P(17)=0.
P(18)=0.
P(19)=0.
P(20)=0.

TIME= 15.0000 AV.NO. IN SYS.= 2.282 STD.DEV.= 2.182

P(0)=	.233000
P(1)=	.219000
P(2)=	.179000
P(3)=	.121000
P(4)=	.093000
P(5)=	.066000
P(6)=	.040000
P(7)=	.020000
P(8)=	.015000
P(9)=	.006000
P(10)=	.005000
P(11)=	.002000
P(12)=0.	
P(13)=0.	
P(14)=	.001000
P(15)=0.	
P(16)=0.	
P(17)=0.	
P(18)=0.	
P(19)=0.	
P(20)=0.	

TIME= 20.0000 AV-NO. IN SYS.= 2.565 STD-DEV.= 2.506

P(0) =	.248000
P(1) =	.182000
P(2) =	.150000
P(3) =	.122000
P(4) =	.095000
P(5) =	.075000
P(6) =	.056000
P(7) =	.045000
P(8) =	.017000
P(9) =	.015000
P(10) =	.005000
P(11) =	.006000
P(12) =	.001000
P(13) =	.001000
P(14) =	.002000
P(15) =	.000000
P(16) =	.000000
P(17) =	.000000
P(18) =	.000000
P(19) =	.000000
P(20) =	.000000

APPENDIX B
LISTING OF BESSEL FUNCTION PROGRAM

The following is a FORTRAN listing and sample output of a program written specifically to obtain the numerical solution of the Bessel function transient analytical expression for the M/M/1 queue (15).

```

FORTRAN IV G LEVEL 20          MAIN          DATE = 72103          18/05/06          PAGE 0001

C
C      SOLUTION OF QUEUEING PROBLEM BY SOLVING BESSEL FUNCTIONS OF
C      THE FIRST KIND. STUART OLSON
C
0001      REAL LAMBDA,MU,P1(500),TITLE(20),EPS(2,0.00005/
0002      CUMUL / %STU/ LAMBDA,MU,ALPHA, SORTA, SORTB, PHRA, PHRB, TERM, EPART,
C      1 COEFF, I
C      NAMELIST /ODATA/ I,N,LAMBDA,MU,IMAX,DT
0003      8 READ (5,1,FND=100) TITLE
0004      1 FORMAT (20A4)
0005      READ (5,2) ODATA,END=100,ERR=101)
0006      ALPHA=LAMBDA/MU
0007      PRINT 5, TITLE, I, N, LAMBDA, MU, ALPHA
0008      5 FORMAT (1H120A4/1H09X6HINIT 011X4HNNX9X6HLAMBDA13X21H10X5HALPHA
0009      1 /1H 2115,3F15.4/1H08X1H13X8HP-VECTOR)
C      LINE=0
C      NSTAR=N
C
C      COMPUTE AUXILIARY CONSTANTS.
C
0012      T=C.
0013      NLIM=7+1
0014      RATIO=MU/LAMBDA
0015      SORTA=SQRT(RATIO)
0016      SORTB=SQRTA*1
0017      SORTB=SQRT(1+.LAMBDA*MU)
0018      GRAM=LAMBDA/MU
0019      PLU=-1/LAMBDA*MU
0020      7 PHRA=SQRTI
0021      TERM=11.-GRM)
0022      COEFF=SQRTI*SORTA
0023      PHRB=COEFF
0024      EPART=EXP(PLUS*T)
0025      YSTAR=C
0026      D) 3 NH=1,NLIM
0027      COEFF=COEFF*SQRTA
0028      PH(NH)=PDFSSQ(NH-1,T)
0029      IF (YSTAR.EQ.0 .AND. PH(NH).LT.EPS) NSTAR=NH
C      TERM=TERM*GRM
C      PHRA=PHRA
C      PHRB=PHRB/SORTA
0032      3 CONTINUE
0033      IF (NSTAR.EQ.0) NSTAR=N
0034      LINE=LINE+(NSTAR+5)/10
0035      IF (LINE.LT.50) GO TO 10
0036      LINE=C
0037      PRINT 5, TITLE, I, N, LAMBDA, MU, ALPHA
0038      10 NNN=MIN(10,NSTAR)
0039

```

```

FORTRAN IV C LEVEL 20          MAIN          DATE = 72103          18/05/06          PAGE 0002

0040      PRINT 11,I,(J,PN(J),J=1,NN*)
0041      11 FORMAT (1H F9.4,10(14,F7.4))
0042      N2=NNSTAR-NN*
0043      IF (NN.LE.0) GO TO 13
0044      N0=NN+1
0045      PRINT 12,(J,PN(J),J=N0,NSTAR)
0046      12 FORMAT (1H 9X10(14,F7.4))
0047      13 I=1+OI
0048      IF (I.LE.IMAX) GO TO 7
0049      GO TO 8
0050      101 PRINT 102
0051      102 FORMAT ('INPUT ERROR')
0052      100 CALL EXIF
0053      END

```

```

FORTRAN IV G LEVEL 20          PBESSQ          DATE = 72103          18/05/06          PAGE 0001

0001      FUNCTION PBESSQ(N,T)
0002      REAL LAMBDA,MU
0003      COMMON /SSSTU/ LAMBDA,MU,ALPHA, SORTA, SORTB, PMRA, PMRB, TERM, EPART,
0004      1 COEFF,I
0005      IF (1-LE.0-0) GO TO 12
0006      SUM=0.
0007      A=SQRT(1+K)
0008      K=1+K+1
0009      KLIM=N+50
0010      CUFF=COEFF
0011      3 K=K+1
0012      COEF=COEF*SORTA
0013      CALL BESIT(X,K,BI,IFR)
0014      IF (IFR.EQ.0) GO TO 101
0015      BI=C.
0016      SUM2=COEF*B1
0017      SUM=SUM+SUM2
0018      IF (.NOT.(SUM2.LT.1.E-6*SUM .OR. K.GT.KLIM)) GO TO 3
0019      2 IN=1-N
0020      CALL PESIT(X,IABS(IN),BI,IER)
0021      IF (IFR.EQ.0) GO TO 103
0022      BI=0.
0023      103 IPN=1+IN+1
0024      CALL BESIT(X,IPN,BIR,IER)
0025      IF (IFR.EQ.0) GO TO 104
0026      BIR=0.
0027      PNT=EPART*(PMRA*B1+PMRB*BIR+TERM*SUM)
0028      PBESSQ=PNT
0029      RETURN
0030      12 PBESSQ=0.0
0031      IF (N.EQ.1) PBESSQ=1.0
0032      END

```

PAGE 0001

18/05/06

DATE = 72103

FORTRAN IV G LEVEL 20

```

0001      SUBROUTINE BES1(X,N,B1,IER)
0002      DATA UNDER/1.E-50/
0003      IER=0
0004      B1=1.
0005      IF (N)150,15,10
0006      10 IF (X)160,20,20
0007      15 IF (X)160,17,20
0008      17 RETURN
0009      C009 IOL=i-6
0010      20 IF (X-12.140,40,30
0011      30 IF (X-FLD(10))40,40,110
0012      40 XX=X/2.
0013      50 IER=1.
0014      IF (N)70,70,55
0015      55 DO 60 I=1,N
0016      FI=1
0017      IF (ABS(TERM)-UNDER)56,60,60
0018      56 IER=3
0019      B1=0.
0020      RETURN
0021      C021 TERM=TERM*XX/FI
0022      70 B1=TERM
0023      XX=XX*XX
0024      DO 70 K=1,1000
0025      IF (ABS(TERM)-ABS(B1*TOL))100,100,80
0026      80 FK=K*(N+K)
0027      TERM=TERM*(XX/FK)
0028      90 B1=B1+TERM
0029      100 RETURN
0030      C030 FX=4*H*H
0031      110 IF (X-170.115,111,111
0032      111 IER=4
0033      RETURN
0034      115 XX=1./18.*X;
0035      TERM=1.
0036      B1=1.
0037      DO 130 K=1,30
0038      IF (ABS(TERM)-ABS(TOL*B1))140,140,120
0039      120 FK=(2*K-1)*2
0040      TERM=TERM*XX*(FK-FN)/FLD(10)
0041      130 B1=B1+TERM
0042      GO TO 40
0043      C043 PI=3.141593
0044      B1=61*EXP(X)/SORT(2.*PI*X)
0045      GO TO 100
0046      150 IER=1
0047      GO TO 100
0048      160 IER=2

```

PAGE 0002

18/05/06

DATE = 72103

FORTRAN IV G LEVEL 20

BES1

```

0049      GO TO 100
0050      END

```

M/M/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART OLSON

	INIT Q	NMAX	LAMBDA	MU	ALPHA
	0	21	0.4500	0.5000	0.9000
T	P-VECTOR				
0.0	1 1.0000	2 0.0	3 0.0000	4 0.0000	5 0.0000
0.1000	1 0.9571	2 0.0420	3 0.0035	4 0.0001	5 0.0000
0.2000	1 0.9179	2 0.0785	3 0.0035	4 0.0001	5 0.0000
0.3000	1 0.8822	2 0.1103	3 0.0072	4 0.0007	5 0.0000
0.4000	1 0.8475	2 0.1379	3 0.0120	4 0.0013	5 0.0001
0.5000	1 0.8195	2 0.1612	3 0.0174	4 0.0020	5 0.0001
0.6000	1 0.7919	2 0.1828	3 0.0233	4 0.0030	5 0.0002
0.7000	1 0.7665	2 0.2009	3 0.0296	4 0.0041	5 0.0004
0.8000	1 0.7431	2 0.2166	3 0.0360	4 0.0055	5 0.0005
0.9000	1 0.7214	2 0.2303	3 0.0426	4 0.0070	5 0.0008
1.0000	1 0.7012	2 0.2421	3 0.0492	4 0.0087	5 0.0010
1.1000	1 0.6825	2 0.2524	3 0.0557	4 0.0105	5 0.0014
1.2000	1 0.6651	2 0.2613	3 0.0622	4 0.0124	5 0.0017
1.3000	1 0.6489	2 0.2697	3 0.0685	4 0.0144	5 0.0022
1.4000	1 0.6337	2 0.2775	3 0.0746	4 0.0166	5 0.0026
1.5000	1 0.6195	2 0.2841	3 0.0805	4 0.0188	5 0.0032
1.6000	1 0.6062	2 0.2899	3 0.0862	4 0.0211	5 0.0038
1.7000	1 0.5936	2 0.2952	3 0.0917	4 0.0234	5 0.0044
1.8000	1 0.5818	2 0.2999	3 0.0970	4 0.0258	5 0.0051
1.9000	1 0.5707	2 0.3045	3 0.1021	4 0.0283	5 0.0058
2.0000	1 0.5602	2 0.3087	3 0.1069	4 0.0306	5 0.0066
2.1000	1 0.5503	2 0.3127	3 0.1115	4 0.0331	5 0.0075
2.2000	1 0.5409	2 0.3163	3 0.1158	4 0.0355	5 0.0083
2.3000	1 0.5319	2 0.3195	3 0.1200	4 0.0380	5 0.0092
2.4000	1 0.5234	2 0.3223	3 0.1239	4 0.0404	5 0.0102
2.5000	1 0.5154	2 0.3252	3 0.1276	4 0.0428	5 0.0111
2.6000	1 0.5077	2 0.3277	3 0.1312	4 0.0452	5 0.0121
2.7000	1 0.5004	2 0.3300	3 0.1345	4 0.0475	5 0.0132
2.8000	1 0.4934	2 0.3320	3 0.1377	4 0.0498	5 0.0142
2.9000	1 0.4867	2 0.3339	3 0.1406	4 0.0521	5 0.0153
3.0000	1 0.4803	2 0.3358	3 0.1435	4 0.0544	5 0.0164
3.1000	1 0.4741	2 0.3374	3 0.1461	4 0.0566	5 0.0175
3.2000	1 0.4682	2 0.3389	3 0.1486	4 0.0588	5 0.0186
3.3000	1 0.4626	2 0.3402	3 0.1510	4 0.0609	5 0.0197
3.4000	1 0.4572	2 0.3413	3 0.1532	4 0.0629	5 0.0209
3.5000	1 0.4519	2 0.3425	3 0.1553	4 0.0650	5 0.0222
3.6000	1 0.4467	2 0.3435	3 0.1571	4 0.0670	5 0.0232
3.7000	1 0.4417	2 0.3441	3 0.1589	4 0.0689	5 0.0243
3.8000	1 0.4367	2 0.3449	3 0.1606	4 0.0708	5 0.0255
3.9000	1 0.4319	2 0.3455	3 0.1626	4 0.0726	5 0.0266
4.0000	1 0.4273	2 0.3460	3 0.1641	4 0.0744	5 0.0278
4.1000	1 0.4228	2 0.3467	3 0.1656	4 0.0762	5 0.0289
4.2000	1 0.4184	2 0.3472	3 0.1669	4 0.0779	5 0.0301
4.3000	1 0.4141	2 0.3477	3 0.1682	4 0.0795	5 0.0312
4.4000	1 0.4099	2 0.3482	3 0.1694	4 0.0811	5 0.0323
4.5000	1 0.4058	2 0.3487	3 0.1705	4 0.0827	5 0.0335
4.6000	1 0.4018	2 0.3492	3 0.1716	4 0.0842	5 0.0346
4.7000	1 0.4018	2 0.3492	3 0.1726	4 0.0842	5 0.0346
4.8000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
4.9000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
5.0000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
5.1000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
5.2000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
5.3000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
5.4000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
5.5000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
5.6000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
5.7000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
5.8000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
5.9000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
6.0000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
6.1000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
6.2000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
6.3000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
6.4000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
6.5000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
6.6000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
6.7000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
6.8000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
6.9000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
7.0000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
7.1000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
7.2000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
7.3000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
7.4000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
7.5000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
7.6000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
7.7000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
7.8000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
7.9000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
8.0000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
8.1000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
8.2000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
8.3000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
8.4000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
8.5000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
8.6000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
8.7000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
8.8000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
8.9000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
9.0000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
9.1000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
9.2000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
9.3000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
9.4000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
9.5000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
9.6000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
9.7000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
9.8000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
9.9000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122
10.0000	2 0.4018	3 0.1726	4 0.0842	5 0.0346	6 0.0122

M/M/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART OLSON

INIT Q	RMAX	LAMBDA	MU	ALPHA	
C	21	0.4500	0.5000	0.5000	
T - P-VECTORS					
7-3000	2 3-2652	3 0-1835	4 0-1111	5 0-0593	6 0-0281
	4 0-1111	5 0-0593	6 0-0281	7 0-0119	8 0-0046
7-4000	2 0-3352	3 0-1836	4 0-1118	5 0-0601	6 0-0287
	4 0-1118	5 0-0601	6 0-0287	7 0-0123	8 0-0047
7-5000	2 0-3334	3 0-1837	4 0-1124	5 0-0608	6 0-0293
	4 0-1124	5 0-0608	6 0-0293	7 0-0126	8 0-0049
7-6000	2 0-3316	3 0-1837	4 0-1130	5 0-0615	6 0-0299
	4 0-1130	5 0-0615	6 0-0299	7 0-0130	8 0-0051
7-7000	2 0-3299	3 0-1838	4 0-1136	5 0-0623	6 0-0304
	4 0-1136	5 0-0623	6 0-0304	7 0-0134	8 0-0053
7-8000	2 0-3283	3 0-1838	4 0-1142	5 0-0630	6 0-0310
	4 0-1142	5 0-0630	6 0-0310	7 0-0137	8 0-0055
7-9000	2 0-3265	3 0-1839	4 0-1148	5 0-0637	6 0-0316
	4 0-1148	5 0-0637	6 0-0316	7 0-0141	8 0-0057
8-0000	2 0-3250	3 0-1838	4 0-1153	5 0-0644	6 0-0322
	4 0-1153	5 0-0644	6 0-0322	7 0-0145	8 0-0059
8-1000	2 0-3234	3 0-1838	4 0-1158	5 0-0650	6 0-0327
	4 0-1158	5 0-0650	6 0-0327	7 0-0148	8 0-0061
8-2000	2 0-3218	3 0-1837	4 0-1163	5 0-0657	6 0-0333
	4 0-1163	5 0-0657	6 0-0333	7 0-0152	8 0-0063
8-3000	2 0-3203	3 0-1837	4 0-1168	5 0-0664	6 0-0338
	4 0-1168	5 0-0664	6 0-0338	7 0-0156	8 0-0065
8-4000	2 0-3188	3 0-1837	4 0-1173	5 0-0670	6 0-0344
	4 0-1173	5 0-0670	6 0-0344	7 0-0159	8 0-0067
8-5000	2 0-3173	3 0-1836	4 0-1178	5 0-0676	6 0-0349
	4 0-1178	5 0-0676	6 0-0349	7 0-0163	8 0-0069
8-6000	2 0-3159	3 0-1835	4 0-1182	5 0-0682	6 0-0355
	4 0-1182	5 0-0682	6 0-0355	7 0-0167	8 0-0071
8-7000	2 0-3144	3 0-1835	4 0-1186	5 0-0688	6 0-0360
	4 0-1186	5 0-0688	6 0-0360	7 0-0170	8 0-0074
8-8000	2 0-3130	3 0-1834	4 0-1191	5 0-0694	6 0-0365
	4 0-1191	5 0-0694	6 0-0365	7 0-0174	8 0-0076
8-9000	2 0-3117	3 0-1833	4 0-1195	5 0-0700	6 0-0370
	4 0-1195	5 0-0700	6 0-0370	7 0-0178	8 0-0078
9-0000	2 0-3103	3 0-1832	4 0-1199	5 0-0706	6 0-0376
	4 0-1199	5 0-0706	6 0-0376	7 0-0182	8 0-0080
9-1000	2 0-3089	3 0-1831	4 0-1203	5 0-0712	6 0-0381
	4 0-1203	5 0-0712	6 0-0381	7 0-0185	8 0-0082
9-2000	2 0-3075	3 0-1830	4 0-1207	5 0-0717	6 0-0386
	4 0-1207	5 0-0717	6 0-0386	7 0-0189	8 0-0084
9-3000	2 0-3061	3 0-1829	4 0-1211	5 0-0722	6 0-0391
	4 0-1211	5 0-0722	6 0-0391	7 0-0192	8 0-0087
9-4000	2 0-3047	3 0-1828	4 0-1215	5 0-0728	6 0-0396
	4 0-1215	5 0-0728	6 0-0396	7 0-0196	8 0-0089
9-5000	2 0-3033	3 0-1827	4 0-1219	5 0-0733	6 0-0401
	4 0-1219	5 0-0733	6 0-0401	7 0-0200	8 0-0091
9-6000	2 0-3019	3 0-1826	4 0-1223	5 0-0738	6 0-0406
	4 0-1223	5 0-0738	6 0-0406	7 0-0205	8 0-0093
9-7000	2 0-3005	3 0-1824	4 0-1227	5 0-0743	6 0-0410
	4 0-1227	5 0-0743	6 0-0410	7 0-0207	8 0-0096
9-8000	2 0-2991	3 0-1823	4 0-1231	5 0-0748	6 0-0415
	4 0-1231	5 0-0748	6 0-0415	7 0-0211	8 0-0100
9-9000	2 0-2977	3 0-1822	4 0-1235	5 0-0753	6 0-0420
	4 0-1235	5 0-0753	6 0-0420	7 0-0214	8 0-0104
10-0000	2 0-2963	3 0-1821	4 0-1239	5 0-0758	6 0-0425
	4 0-1239	5 0-0758	6 0-0425	7 0-0217	8 0-0108
10-1000	2 0-2949	3 0-1820	4 0-1243	5 0-0763	6 0-0430
	4 0-1243	5 0-0763	6 0-0430	7 0-0220	8 0-0112
10-2000	2 0-2935	3 0-1819	4 0-1247	5 0-0768	6 0-0435
	4 0-1247	5 0-0768	6 0-0435	7 0-0223	8 0-0115
10-3000	2 0-2921	3 0-1818	4 0-1251	5 0-0773	6 0-0440
	4 0-1251	5 0-0773	6 0-0440	7 0-0226	8 0-0118
10-4000	2 0-2907	3 0-1817	4 0-1255	5 0-0778	6 0-0445
	4 0-1255	5 0-0778	6 0-0445	7 0-0229	8 0-0121
10-5000	2 0-2893	3 0-1816	4 0-1259	5 0-0783	6 0-0450
	4 0-1259	5 0-0783	6 0-0450	7 0-0232	8 0-0124
10-6000	2 0-2879	3 0-1815	4 0-1263	5 0-0788	6 0-0455
	4 0-1263	5 0-0788	6 0-0455	7 0-0235	8 0-0127
10-7000	2 0-2865	3 0-1814	4 0-1267	5 0-0793	6 0-0460
	4 0-1267	5 0-0793	6 0-0460	7 0-0238	8 0-0130
10-8000	2 0-2851	3 0-1813	4 0-1271	5 0-0798	6 0-0465
	4 0-1271	5 0-0798	6 0-0465	7 0-0241	8 0-0133
10-9000	2 0-2837	3 0-1812	4 0-1275	5 0-0803	6 0-0470
	4 0-1275	5 0-0803	6 0-0470	7 0-0244	8 0-0136
11-0000	2 0-2823	3 0-1811	4 0-1279	5 0-0808	6 0-0475
	4 0-1279	5 0-0808	6 0-0475	7 0-0247	8 0-0139
11-1000	2 0-2809	3 0-1810	4 0-1283	5 0-0813	6 0-0480
	4 0-1283	5 0-0813	6 0-0480	7 0-0250	8 0-0142
11-2000	2 0-2795	3 0-1809	4 0-1287	5 0-0818	6 0-0485
	4 0-1287	5 0-0818	6 0-0485	7 0-0253	8 0-0145
11-3000	2 0-2781	3 0-1808	4 0-1291	5 0-0823	6 0-0490
	4 0-1291	5 0-0823	6 0-0490	7 0-0256	8 0-0148
11-4000	2 0-2767	3 0-1807	4 0-1295	5 0-0828	6 0-0495
	4 0-1295	5 0-0828	6 0-0495	7 0-0259	8 0-0151
11-5000	2 0-2753	3 0-1806	4 0-1299	5 0-0833	6 0-0500
	4 0-1299	5 0-0833	6 0-0500	7 0-0262	8 0-0154
11-6000	2 0-2739	3 0-1805	4 0-1303	5 0-0838	6 0-0505
	4 0-1303	5 0-0838	6 0-0505	7 0-0265	8 0-0157
11-7000	2 0-2725	3 0-1804	4 0-1307	5 0-0843	6 0-0510
	4 0-1307	5 0-0843	6 0-0510	7 0-0268	8 0-0160
11-8000	2 0-2711	3 0-1803	4 0-1311	5 0-0848	6 0-0515
	4 0-1311	5 0-0848	6 0-0515	7 0-0271	8 0-0163
11-9000	2 0-2697	3 0-1802	4 0-1315	5 0-0853	6 0-0520
	4 0-1315	5 0-0853	6 0-0520	7 0-0274	8 0-0166
12-0000	2 0-2683	3 0-1801	4 0-1319	5 0-0858	6 0-0525
	4 0-1319	5 0-0858	6 0-0525	7 0-0277	8 0-0169
12-1000	2 0-2669	3 0-1800	4 0-1323	5 0-0863	6 0-0530
	4 0-1323	5 0-0863	6 0-0530	7 0-0280	8 0-0172
12-2000	2 0-2655	3 0-1799	4 0-1327	5 0-0868	6 0-0535
	4 0-1327	5 0-0868	6 0-0535	7 0-0283	8 0-0175
12-3000	2 0-2641	3 0-1798	4 0-1331	5 0-0873	6 0-0540
	4 0-1331	5 0-0873	6 0-0540	7 0-0286	8 0-0178
12-4000	2 0-2627	3 0-1797	4 0-1335	5 0-0878	6 0-0545
	4 0-1335	5 0-0878	6 0-0545	7 0-0289	8 0-0181
12-5000	2 0-2613	3 0-1796	4 0-1339	5 0-0883	6 0-0550
	4 0-1339	5 0-0883	6 0-0550	7 0-0292	8 0-0184
12-6000	2 0-2599	3 0-1795	4 0-1343	5 0-0888	6 0-0555
	4 0-1343	5 0-0888	6 0-0555	7 0-0295	8 0-0187
12-7000	2 0-2585	3 0-1794	4 0-1347	5 0-0893	6 0-0560
	4 0-1347	5 0-0893	6 0-0560	7 0-0298	8 0-0190
12-8000	2 0-2571	3 0-1793	4 0-1351	5 0-0898	6 0-0565
	4 0-1351	5 0-0898	6 0-0565	7 0-0301	8 0-0193
12-9000	2 0-2557	3 0-1792	4 0-1355	5 0-0903	6 0-0570
	4 0-1355	5 0-0903	6 0-0570	7 0-0304	8 0-0196
13-0000	2 0-2543	3 0-1791	4 0-1359	5 0-0908	6 0-0575
	4 0-1359	5 0-0908	6 0-0575	7 0-0307	8 0-0199
13-1000	2 0-2529	3 0-1790	4 0-1363	5 0-0913	6 0-0580
	4 0-1363	5 0-0913	6 0-0580	7 0-0310	8 0-0202
13-2000	2 0-2515	3 0-1789	4 0-1367	5 0-0918	6 0-0585
	4 0-1367	5 0-0918	6 0-0585	7 0-0313	8 0-0205
13-3000	2 0-2501	3 0-1788	4 0-1371	5 0-0923	6 0-0590
	4 0-1371	5 0-0923	6 0-0590	7 0-0316	8 0-0208
13-4000	2 0-2487	3 0-1787	4 0-1375	5 0-0928	6 0-0595
	4 0-1375	5 0-0928	6 0-0595	7 0-0319	8 0-0211
13-5000	2 0-2473	3 0-1786	4 0-1379	5 0-0933	6 0-0600
	4 0-1379	5 0-0933	6 0-0600	7 0-0322	8 0-0214
13-6000	2 0-2459	3 0-1785	4 0-1383	5 0-0938	6 0-0605
	4 0-1383	5 0-0938	6 0-0605	7 0-0325	8 0-0217
13-7000	2 0-2445	3 0-1784	4 0-1387	5 0-0943	6 0-0610
	4 0-1387	5 0-0943	6 0-0610	7 0-0328	8 0-0220
13-8000	2 0-2431	3 0-1783	4 0-1391	5 0-0948	6 0-0615
	4 0-1391	5 0-0948	6 0-0615	7 0-0331	8 0-0223
13-9000	2 0-2417	3 0-1782	4 0-1395	5 0-0953	6 0-0620
	4 0-1395	5 0-0953	6 0-0620	7 0-0334	8 0-0226
14-0000	2 0-2403	3 0-1781	4 0-1399	5 0-0958	6 0-0625
	4 0-1399	5 0-0958	6 0-0625	7 0-0337	8 0-0229
14-1000	2 0-2389	3 0-1780	4 0-1403	5 0-0963	6 0-0630
	4 0-1403	5 0-0963	6 0-0630	7 0-0340	8 0-0232
14-2000	2 0-2375	3 0-1779	4 0-1407	5 0-0968	6 0-0635
	4 0-1407	5 0-0968	6 0-0635	7 0-0343	8 0-0235
14-3000	2 0-2361	3 0-1778	4 0-1411	5 0-0973	6 0-0640
	4 0-1411	5 0-0973	6 0-0640	7 0-0346	8 0-0238
14-4000	2 0-2347	3 0-1777	4 0-1415	5 0-0978	6 0-0645
	4 0-1415	5 0-0978	6 0-0645	7 0-0349	8 0-0241
14-5000	2 0-2333	3 0-1776	4 0-1419	5 0-0983	6 0-0650
	4 0-1419	5 0-0983	6 0-0650	7 0-0352	8 0-0244
14-6000	2 0-2319	3 0-1775	4 0-1423	5 0-0988	6 0-0655
	4 0-1423	5 0-0988	6 0-0655	7 0-0355	8 0-0247
14-7000	2 0-2305	3 0-1774	4 0-1427	5 0-0993	6 0-0660
	4 0-1427	5 0-0993	6 0-0660	7 0-0358	8 0-0250
14-8000	2 0-2291	3 0-1773	4 0-1431	5 0-0998	6 0-0665
	4 0-1431	5 0-0998	6 0-0665	7 0-0361	8 0-0253
14-9000	2 0-2277	3 0-1772	4 0-1435	5 0-1003	6 0-0670
	4 0-1435	5 0-1003	6 0-0670</		

M/K/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART ULSON

T	INIT Q	PMAX	LAMBDA	MU	ALPHA	
	0	21	0.4500	0.5000	0.9000	
12.2999	P-VECTUR	2	0.1780	5	0.0865	5
12.3999	1 0.2730	2 0.2278	3 0.1780	5 0.0865	5 0.0518	7 0.0295
12.4999	6 0.6518	7 0.6235	8 0.0156	10 0.0035	11 0.0015	12 0.0006
12.5999	1 0.2742	2 0.2273	3 0.1778	5 0.0848	6 0.0522	7 0.0298
12.6999	6 0.6522	7 0.6248	8 0.0158	10 0.0036	11 0.0016	12 0.0006
12.7999	1 0.2733	2 0.2287	3 0.1776	5 0.0851	6 0.0525	7 0.0301
12.8999	6 0.6525	7 0.6301	8 0.0160	10 0.0037	11 0.0016	12 0.0003
12.9999	1 0.2725	2 0.2282	3 0.1774	5 0.0854	6 0.0524	7 0.0304
13.0999	6 0.6529	7 0.6304	8 0.0163	10 0.0038	11 0.0017	12 0.0007
13.1999	1 0.2717	2 0.2276	3 0.1772	5 0.0857	6 0.0532	7 0.0307
13.2999	6 0.6537	7 0.6307	8 0.0165	10 0.0039	11 0.0017	12 0.0007
13.3999	1 0.2709	2 0.2271	3 0.1771	5 0.0859	6 0.0535	7 0.0310
13.4999	6 0.6535	7 0.6310	8 0.0167	10 0.0040	11 0.0018	12 0.0007
13.5999	1 0.2711	2 0.2266	3 0.1767	5 0.0862	6 0.0539	7 0.0313
13.6999	6 0.6537	7 0.6313	8 0.0167	10 0.0041	11 0.0018	12 0.0008
13.7999	1 0.2693	2 0.2261	3 0.1767	5 0.0865	6 0.0542	7 0.0316
13.8999	6 0.6542	7 0.6316	8 0.0172	10 0.0041	11 0.0018	12 0.0008
13.9999	1 0.2694	2 0.2265	3 0.1765	5 0.0867	6 0.0545	7 0.0319
14.0999	6 0.6545	7 0.6319	8 0.0174	10 0.0042	11 0.0019	12 0.0008
14.1999	1 0.2677	2 0.2259	3 0.1763	5 0.0870	6 0.0548	7 0.0322
14.2999	6 0.6547	7 0.6322	8 0.0176	10 0.0043	11 0.0019	12 0.0008
14.3999	1 0.2675	2 0.2245	3 0.1761	5 0.0873	6 0.0551	7 0.0325
14.4999	6 0.6551	7 0.6325	8 0.0178	10 0.0044	11 0.0020	12 0.0009
14.5999	1 0.2662	2 0.2240	3 0.1759	5 0.0875	6 0.0554	7 0.0327
14.6999	6 0.6554	7 0.6327	8 0.0181	10 0.0045	11 0.0021	12 0.0009
14.7999	1 0.2655	2 0.2235	3 0.1757	5 0.0877	6 0.0557	7 0.0330
14.8999	6 0.6560	7 0.6332	8 0.0176	10 0.0046	11 0.0021	12 0.0009
14.9999	1 0.2647	2 0.2230	3 0.1755	5 0.0880	6 0.0560	7 0.0333
15.0999	6 0.6567	7 0.6335	8 0.0179	10 0.0047	11 0.0022	12 0.0009
15.1999	1 0.2633	2 0.2225	3 0.1753	5 0.0882	6 0.0563	7 0.0336
15.2999	6 0.6573	7 0.6341	8 0.0179	10 0.0048	11 0.0022	12 0.0009
15.3999	1 0.2633	2 0.2220	3 0.1751	5 0.0884	6 0.0566	7 0.0339
15.4999	6 0.6577	7 0.6344	8 0.0180	10 0.0049	11 0.0023	12 0.0009
15.5999	1 0.2626	2 0.2215	3 0.1743	5 0.0887	6 0.0569	7 0.0342
15.6999	6 0.6582	7 0.6347	8 0.0181	10 0.0050	11 0.0023	12 0.0009
15.7999	1 0.2619	2 0.2210	3 0.1747	5 0.0889	6 0.0572	7 0.0344
15.8999	6 0.6584	7 0.6349	8 0.0182	10 0.0051	11 0.0024	12 0.0009
15.9999	1 0.2612	2 0.2206	3 0.1745	5 0.0891	6 0.0575	7 0.0347
16.0999	6 0.6587	7 0.6350	8 0.0184	10 0.0052	11 0.0024	12 0.0009
16.1999	1 0.2605	2 0.2201	3 0.1743	5 0.0893	6 0.0578	7 0.0350
16.2999	6 0.6590	7 0.6353	8 0.0185	10 0.0053	11 0.0025	12 0.0009
16.3999	1 0.2598	2 0.2196	3 0.1741	5 0.0895	6 0.0581	7 0.0352
16.4999	6 0.6592	7 0.6355	8 0.0187	10 0.0054	11 0.0025	12 0.0009
16.5999	1 0.2591	2 0.2191	3 0.1739	5 0.0897	6 0.0583	7 0.0355
16.6999	6 0.6595	7 0.6357	8 0.0188	10 0.0055	11 0.0026	12 0.0009
16.7999	1 0.2584	2 0.2187	3 0.1737	5 0.0899	6 0.0586	7 0.0358
16.8999	6 0.6598	7 0.6359	8 0.0189	10 0.0056	11 0.0027	12 0.0009
16.9999	1 0.2577	2 0.2182	3 0.1735	5 0.0901	6 0.0589	7 0.0360
17.0999	6 0.6600	7 0.6360	8 0.0190	10 0.0057	11 0.0027	12 0.0009
17.1999	1 0.2571	2 0.2178	3 0.1733	5 0.0903	6 0.0591	7 0.0363
17.2999	6 0.6603	7 0.6363	8 0.0191	10 0.0058	11 0.0028	12 0.0009
17.3999	1 0.2564	2 0.2173	3 0.1731	5 0.0905	6 0.0593	7 0.0365
17.4999	6 0.6606	7 0.6366	8 0.0192	10 0.0059	11 0.0029	12 0.0009
17.5999	1 0.2557	2 0.2168	3 0.1729	5 0.0907	6 0.0595	7 0.0367
17.6999	6 0.6609	7 0.6369	8 0.0193	10 0.0060	11 0.0030	12 0.0009
17.7999	1 0.2550	2 0.2163	3 0.1727	5 0.0909	6 0.0597	7 0.0369
17.8999	6 0.6612	7 0.6372	8 0.0194	10 0.0061	11 0.0031	12 0.0009
17.9999	1 0.2543	2 0.2158	3 0.1725	5 0.0911	6 0.0599	7 0.0371
18.0999	6 0.6615	7 0.6375	8 0.0195	10 0.0062	11 0.0032	12 0.0009
18.1999	1 0.2536	2 0.2153	3 0.1723	5 0.0913	6 0.0601	7 0.0373
18.2999	6 0.6618	7 0.6378	8 0.0196	10 0.0063	11 0.0033	12 0.0009
18.3999	1 0.2529	2 0.2148	3 0.1721	5 0.0915	6 0.0603	7 0.0375
18.4999	6 0.6621	7 0.6381	8 0.0197	10 0.0064	11 0.0034	12 0.0009
18.5999	1 0.2522	2 0.2143	3 0.1719	5 0.0917	6 0.0605	7 0.0377
18.6999	6 0.6624	7 0.6384	8 0.0198	10 0.0065	11 0.0035	12 0.0009
18.7999	1 0.2515	2 0.2138	3 0.1717	5 0.0919	6 0.0607	7 0.0379
18.8999	6 0.6627	7 0.6387	8 0.0199	10 0.0066	11 0.0036	12 0.0009
18.9999	1 0.2508	2 0.2133	3 0.1715	5 0.0921	6 0.0609	7 0.0381
19.0999	6 0.6630	7 0.6390	8 0.0200	10 0.0067	11 0.0037	12 0.0009
19.1999	1 0.2501	2 0.2128	3 0.1713	5 0.0923	6 0.0611	7 0.0383
19.2999	6 0.6633	7 0.6393	8 0.0201	10 0.0068	11 0.0038	12 0.0009
19.3999	1 0.2494	2 0.2123	3 0.1711	5 0.0925	6 0.0613	7 0.0385
19.4999	6 0.6636	7 0.6396	8 0.0202	10 0.0069	11 0.0039	12 0.0009
19.5999	1 0.2487	2 0.2118	3 0.1709	5 0.0927	6 0.0615	7 0.0387
19.6999	6 0.6639	7 0.6399	8 0.0203	10 0.0070	11 0.0040	12 0.0009
19.7999	1 0.2480	2 0.2113	3 0.1707	5 0.0929	6 0.0617	7 0.0389
19.8999	6 0.6642	7 0.6402	8 0.0204	10 0.0071	11 0.0041	12 0.0009
19.9999	1 0.2473	2 0.2108	3 0.1705	5 0.0931	6 0.0619	7 0.0391
20.0999	6 0.6645	7 0.6405	8 0.0205	10 0.0072	11 0.0042	12 0.0009
20.1999	1 0.2466	2 0.2103	3 0.1703	5 0.0933	6 0.0621	7 0.0393
20.2999	6 0.6648	7 0.6408	8 0.0206	10 0.0073	11 0.0043	12 0.0009
20.3999	1 0.2459	2 0.2098	3 0.1701	5 0.0935	6 0.0623	7 0.0395
20.4999	6 0.6651	7 0.6411	8 0.0207	10 0.0074	11 0.0044	12 0.0009
20.5999	1 0.2452	2 0.2093	3 0.1699	5 0.0937	6 0.0625	7 0.0397
20.6999	6 0.6654	7 0.6414	8 0.0208	10 0.0075	11 0.0045	12 0.0009
20.7999	1 0.2445	2 0.2088	3 0.1697	5 0.0939	6 0.0627	7 0.0399
20.8999	6 0.6657	7 0.6417	8 0.0209	10 0.0076	11 0.0046	12 0.0009
20.9999	1 0.2438	2 0.2083	3 0.1695	5 0.0941	6 0.0629	7 0.0401
21.0999	6 0.6660	7 0.6420	8 0.0210	10 0.0077	11 0.0047	12 0.0009
21.1999	1 0.2431	2 0.2078	3 0.1693	5 0.0943	6 0.0631	7 0.0403
21.2999	6 0.6663	7 0.6423	8 0.0211	10 0.0078	11 0.0048	12 0.0009
21.3999	1 0.2424	2 0.2073	3 0.1691	5 0.0945	6 0.0633	7 0.0405
21.4999	6 0.6666	7 0.6426	8 0.0212	10 0.0079	11 0.0049	12 0.0009
21.5999	1 0.2417	2 0.2068	3 0.1689	5 0.0947	6 0.0635	7 0.0407
21.6999	6 0.6669	7 0.6429	8 0.0213	10 0.0080	11 0.0050	12 0.0009
21.7999	1 0.2410	2 0.2063	3 0.1687	5 0.0949	6 0.0637	7 0.0409
21.8999	6 0.6672	7 0.6432	8 0.0214	10 0.0081	11 0.0051	12 0.0009
21.9999	1 0.2403	2 0.2058	3 0.1685	5 0.0951	6 0.0639	7 0.0411
22.0999	6 0.6675	7 0.6435	8 0.0215	10 0.0082	11 0.0052	12 0.0009
22.1999	1 0.2396	2 0.2053	3 0.1683	5 0.0953	6 0.0641	7 0.0413
22.2999	6 0.6678	7 0.6438	8 0.0216	10 0.0083	11 0.0053	12 0.0009
22.3999	1 0.2389	2 0.2048	3 0.1681	5 0.0955	6 0.0643	7 0.0415
22.4999	6 0.6681	7 0.6441	8 0.0217	10 0.0084	11 0.0054	12 0.0009
22.5999	1 0.2382	2 0.2043	3 0.1679	5 0.0957	6 0.0645	7 0.0417
22.6999	6 0.6684	7 0.6444	8 0.0218	10 0.0085	11 0.0055	12 0.0009
22.7999	1 0.2375	2 0.2038	3 0.1677	5 0.0959	6 0.0647	7 0.0419
22.8999	6 0.6687	7 0.6447	8 0.0219	10 0.0086	11 0.0056	12 0.0009
22.9999	1 0.2368	2 0.2033	3 0.1675	5 0.0961	6 0.0649	7 0.0421
23.0999	6 0.6690	7 0.6450	8 0.0220	10 0.0087	11 0.0057	12 0.0009
23.1999	1 0.2361	2 0.2028	3 0.1673	5 0.0963	6 0.0651	7 0.0423
23.2999	6 0.6693	7 0.6453	8 0.0221	10 0.0088	11 0.0058	12 0.0009
23.3999	1 0.2354	2 0.2023	3 0.1671	5 0.0965	6 0.0653	7 0.0425
23.4999	6 0.6696	7 0.6456	8 0.0222	10 0.0089	11 0.0059	12 0.0009
23.5999	1 0.2347	2 0.2018	3 0.1669	5 0.0967	6 0.0655	7 0.0427
23.6999	6 0.6699	7 0.6459	8 0.0223	10 0.0090	11 0.0060	12 0.0009
23.7999	1 0.2340	2 0.2013	3 0.1667	5 0.0969	6 0.0657	7 0.0429
23.8999	6 0.6702	7 0.6462	8 0.0224	10 0.0091	11 0.0061	12 0.0009
23.9999	1 0.2333	2 0.2008	3 0.1665	5 0.0971	6 0.0659	7 0.0431
24.0999	6 0.6705	7 0.6465	8 0.0225	10 0.0092	11 0.0062	12 0.0009
24.1999	1 0.2326	2 0.2003	3 0.1663	5 0.0973	6 0.0661	7 0.0433
24.2999	6 0.6708	7 0.6468	8 0.0226	10 0.0093	11 0.0063	12 0.0009
24.3999	1 0.2319	2 0.1998	3 0.1661	5 0.0975	6 0.0663	7 0.0435
24.4999	6 0.6711	7 0.6471	8 0.0227	10 0.0094	11 0.0064	12 0.0009
24.5999	1 0.2312	2 0.1993	3 0.1659	5 0.0977	6 0.0665	7 0.0437
24.6999	6 0.6714	7 0.6474	8 0.0228	10 0.0095	11 0.0065	12 0.0009
24						

M/M/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART OLSON

	INIT Q	MAX	LAMBDA	MU	ALPHA	
	Q	Z1	0.4500	0.5000	0.9000	
	P-VECTOR					
14.7999	1 0.2565	2 0.2173	3 0.1731	4 0.1293	5 0.0905	6 0.0594
	7 0.0366	8 0.0211	9 0.0115	10 0.0059	11 0.0028	12 0.0013
14.8999	1 0.2558	2 0.2169	3 0.1729	4 0.1293	5 0.0907	6 0.0597
	7 0.0368	8 0.0213	9 0.0116	10 0.0060	11 0.0029	12 0.0013
14.9999	1 0.2552	2 0.2164	3 0.1727	4 0.1294	5 0.0909	6 0.0599
	7 0.0371	8 0.0215	9 0.0118	10 0.0062	11 0.0030	12 0.0014
15.0999	1 0.2546	2 0.2160	3 0.1725	4 0.1294	5 0.0911	6 0.0602
	7 0.0373	8 0.0218	9 0.0119	10 0.0062	11 0.0030	12 0.0014
15.1999	1 0.2539	2 0.2155	3 0.1723	4 0.1294	5 0.0913	6 0.0604
	7 0.0376	8 0.0220	9 0.0121	10 0.0063	11 0.0031	12 0.0014
15.2999	1 0.2533	2 0.2151	3 0.1721	4 0.1294	5 0.0914	6 0.0606
	7 0.0378	8 0.0222	9 0.0122	10 0.0064	11 0.0031	12 0.0015
15.3999	1 0.2527	2 0.2147	3 0.1719	4 0.1294	5 0.0916	6 0.0609
	7 0.0381	8 0.0224	9 0.0124	10 0.0065	11 0.0032	12 0.0015
15.4999	1 0.2521	2 0.2143	3 0.1717	4 0.1295	5 0.0918	6 0.0611
	7 0.0383	8 0.0226	9 0.0125	10 0.0066	11 0.0033	12 0.0015
15.5999	1 0.2515	2 0.2138	3 0.1715	4 0.1295	5 0.0919	6 0.0614
	7 0.0385	8 0.0228	9 0.0127	10 0.0067	11 0.0033	12 0.0016
15.6999	1 0.2509	2 0.2134	3 0.1713	4 0.1295	5 0.0921	6 0.0616
	7 0.0388	8 0.0230	9 0.0129	10 0.0068	11 0.0034	12 0.0016
15.7999	1 0.2503	2 0.2130	3 0.1711	4 0.1295	5 0.0922	6 0.0618
	7 0.0390	8 0.0232	9 0.0130	10 0.0069	11 0.0034	12 0.0016
15.8999	1 0.2497	2 0.2126	3 0.1709	4 0.1295	5 0.0924	6 0.0621
	7 0.0392	8 0.0234	9 0.0132	10 0.0070	11 0.0034	12 0.0017
15.9999	1 0.2492	2 0.2122	3 0.1707	4 0.1295	5 0.0925	6 0.0623
	7 0.0393	8 0.0236	9 0.0133	10 0.0071	11 0.0035	12 0.0017
16.0999	1 0.2486	2 0.2118	3 0.1705	4 0.1295	5 0.0927	6 0.0625
	7 0.0397	8 0.0238	9 0.0135	10 0.0072	11 0.0035	12 0.0017
16.1999	1 0.2480	2 0.2114	3 0.1703	4 0.1295	5 0.0928	6 0.0627
	7 0.0399	8 0.0240	9 0.0136	10 0.0073	11 0.0035	12 0.0018
16.2999	1 0.2475	2 0.2110	3 0.1701	4 0.1295	5 0.0930	6 0.0629
	7 0.0402	8 0.0242	9 0.0138	10 0.0074	11 0.0036	12 0.0018
16.3999	1 0.2469	2 0.2106	3 0.1699	4 0.1295	5 0.0931	6 0.0631
	7 0.0404	8 0.0244	9 0.0139	10 0.0075	11 0.0036	12 0.0019
16.4999	1 0.2463	2 0.2102	3 0.1698	4 0.1295	5 0.0933	6 0.0634
	7 0.0406	8 0.0246	9 0.0141	10 0.0076	11 0.0036	12 0.0019
16.5999	1 0.2458	2 0.2098	3 0.1696	4 0.1295	5 0.0934	6 0.0636
	7 0.0408	8 0.0248	9 0.0142	10 0.0077	11 0.0037	12 0.0020
16.6998	1 0.2453	2 0.2094	3 0.1694	4 0.1295	5 0.0935	6 0.0638
	7 0.0411	8 0.0250	9 0.0144	10 0.0078	11 0.0037	12 0.0020
16.7998	1 0.2447	2 0.2090	3 0.1692	4 0.1295	5 0.0937	6 0.0640
	7 0.0413	8 0.0252	9 0.0145	10 0.0079	11 0.0038	12 0.0021
16.8998	1 0.2442	2 0.2086	3 0.1690	4 0.1295	5 0.0938	6 0.0642
	7 0.0415	8 0.0254	9 0.0147	10 0.0080	11 0.0038	12 0.0021
16.9998	1 0.2437	2 0.2082	3 0.1688	4 0.1295	5 0.0939	6 0.0644
	7 0.0417	8 0.0255	9 0.0148	10 0.0081	11 0.0039	12 0.0021
17.0998	1 0.2431	2 0.2079	3 0.1686	4 0.1295	5 0.0940	6 0.0646
	7 0.0419	8 0.0257	9 0.0150	10 0.0082	11 0.0040	12 0.0022
17.1998	1 0.2426	2 0.2075	3 0.1684	4 0.1295	5 0.0942	6 0.0648
	7 0.0421	8 0.0259	9 0.0151	10 0.0084	11 0.0041	12 0.0022
0 0.0259						

M/M/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS STUART OLSON

	INIT Q	NMAX	LAMBDA	MU	ALPHA	
	0	21	0.4500	0.5000	0.9000	
17.2998	1 0.2421	2 0.2071	3 0.1682	4 0.1294	5 0.0943	6 0.0649
17.2998	8 0.0261	9 0.0153	10 0.0085	11 0.0045	12 0.0022	13 0.0011
17.2998	1 0.2416	2 0.2067	3 0.1680	4 0.1294	5 0.0944	6 0.0651
17.2998	8 0.0263	9 0.0154	10 0.0086	11 0.0045	12 0.0023	13 0.0011
17.2998	1 0.2411	2 0.2064	3 0.1679	4 0.1294	5 0.0945	6 0.0653
17.2998	8 0.0265	9 0.0155	10 0.0087	11 0.0046	12 0.0023	13 0.0011
17.2998	1 0.2406	2 0.2060	3 0.1677	4 0.1294	5 0.0946	6 0.0655
17.2998	8 0.0267	9 0.0157	10 0.0088	11 0.0047	12 0.0024	13 0.0011
17.2998	1 0.2401	2 0.2057	3 0.1675	4 0.1294	5 0.0947	6 0.0657
17.2998	8 0.0269	9 0.0159	10 0.0089	11 0.0047	12 0.0024	13 0.0012
17.2998	1 0.2396	2 0.2053	3 0.1673	4 0.1294	5 0.0948	6 0.0659
17.2998	8 0.0270	9 0.0160	10 0.0090	11 0.0048	12 0.0024	13 0.0012
17.2998	1 0.2391	2 0.2049	3 0.1671	4 0.1293	5 0.0949	6 0.0660
17.2998	8 0.0272	9 0.0161	10 0.0091	11 0.0049	12 0.0025	13 0.0012
17.2998	1 0.2386	2 0.2044	3 0.1669	4 0.1293	5 0.0950	6 0.0662
17.2998	8 0.0274	9 0.0163	10 0.0092	11 0.0049	12 0.0025	13 0.0012
17.2998	1 0.2381	2 0.2042	3 0.1667	4 0.1293	5 0.0951	6 0.0664
17.2998	8 0.0276	9 0.0164	10 0.0093	11 0.0050	12 0.0026	13 0.0013
17.2998	1 0.2377	2 0.2039	3 0.1666	4 0.1293	5 0.0952	6 0.0666
17.2998	8 0.0278	9 0.0166	10 0.0094	11 0.0051	12 0.0026	13 0.0013
17.2998	1 0.2372	2 0.2035	3 0.1664	4 0.1292	5 0.0953	6 0.0667
17.2998	8 0.0279	9 0.0167	10 0.0095	11 0.0052	12 0.0027	13 0.0013
17.2998	1 0.2367	2 0.2032	3 0.1662	4 0.1292	5 0.0954	6 0.0669
17.2998	8 0.0281	9 0.0169	10 0.0096	11 0.0053	12 0.0027	13 0.0013
17.2998	1 0.2363	2 0.2029	3 0.1660	4 0.1292	5 0.0955	6 0.0671
17.2998	8 0.0283	9 0.0170	10 0.0097	11 0.0054	12 0.0028	13 0.0014
17.2998	1 0.2358	2 0.2025	3 0.1658	4 0.1292	5 0.0956	6 0.0672
17.2998	8 0.0285	9 0.0172	10 0.0098	11 0.0054	12 0.0028	13 0.0014
17.2998	1 0.2354	2 0.2022	3 0.1656	4 0.1291	5 0.0957	6 0.0674
17.2998	8 0.0287	9 0.0173	10 0.0099	11 0.0055	12 0.0029	13 0.0014
17.2998	1 0.2349	2 0.2019	3 0.1655	4 0.1291	5 0.0958	6 0.0675
17.2998	8 0.0289	9 0.0174	10 0.0100	11 0.0055	12 0.0029	13 0.0014
17.2998	1 0.2344	2 0.2015	3 0.1653	4 0.1291	5 0.0959	6 0.0677
17.2998	8 0.0291	9 0.0175	10 0.0101	11 0.0056	12 0.0030	13 0.0015
17.2998	1 0.2340	2 0.2012	3 0.1651	4 0.1291	5 0.0960	6 0.0678
17.2998	8 0.0293	9 0.0177	10 0.0102	11 0.0056	12 0.0030	13 0.0015
17.2998	1 0.2335	2 0.2009	3 0.1649	4 0.1290	5 0.0961	6 0.0680
17.2998	8 0.0295	9 0.0178	10 0.0103	11 0.0057	12 0.0031	13 0.0016
17.2998	1 0.2331	2 0.2006	3 0.1647	4 0.1290	5 0.0962	6 0.0681
17.2998	8 0.0297	9 0.0179	10 0.0104	11 0.0057	12 0.0031	13 0.0016
17.2998	1 0.2327	2 0.2003	3 0.1645	4 0.1289	5 0.0963	6 0.0683
17.2998	8 0.0299	9 0.0180	10 0.0105	11 0.0058	12 0.0032	13 0.0016
17.2998	1 0.2322	2 0.2000	3 0.1643	4 0.1289	5 0.0964	6 0.0684
17.2998	8 0.0301	9 0.0181	10 0.0106	11 0.0058	12 0.0032	13 0.0016
17.2998	1 0.2318	2 0.1997	3 0.1641	4 0.1288	5 0.0965	6 0.0686
17.2998	8 0.0303	9 0.0182	10 0.0107	11 0.0059	12 0.0033	13 0.0017
17.2998	1 0.2314	2 0.1994	3 0.1639	4 0.1288	5 0.0966	6 0.0687
17.2998	8 0.0305	9 0.0183	10 0.0108	11 0.0059	12 0.0033	13 0.0017
17.2998	1 0.2310	2 0.1991	3 0.1637	4 0.1287	5 0.0967	6 0.0689
17.2998	8 0.0307	9 0.0184	10 0.0109	11 0.0060	12 0.0034	13 0.0018
17.2998	1 0.2306	2 0.1988	3 0.1635	4 0.1287	5 0.0968	6 0.0690
17.2998	8 0.0309	9 0.0185	10 0.0110	11 0.0060	12 0.0034	13 0.0018
17.2998	1 0.2302	2 0.1986	3 0.1633	4 0.1286	5 0.0969	6 0.0691
17.2998	8 0.0311	9 0.0186	10 0.0111	11 0.0061	12 0.0035	13 0.0019
17.2998	1 0.2298	2 0.1983	3 0.1631	4 0.1286	5 0.0970	6 0.0692
17.2998	8 0.0313	9 0.0187	10 0.0112	11 0.0061	12 0.0035	13 0.0019
17.2998	1 0.2294	2 0.1980	3 0.1629	4 0.1285	5 0.0971	6 0.0693
17.2998	8 0.0315	9 0.0188	10 0.0113	11 0.0062	12 0.0036	13 0.0020
17.2998	1 0.2290	2 0.1977	3 0.1627	4 0.1285	5 0.0972	6 0.0694
17.2998	8 0.0317	9 0.0189	10 0.0114	11 0.0062	12 0.0036	13 0.0020
17.2998	1 0.2286	2 0.1974	3 0.1625	4 0.1284	5 0.0973	6 0.0695
17.2998	8 0.0319	9 0.0190	10 0.0115	11 0.0063	12 0.0037	13 0.0021
17.2998	1 0.2282	2 0.1971	3 0.1623	4 0.1284	5 0.0974	6 0.0696
17.2998	8 0.0321	9 0.0191	10 0.0116	11 0.0063	12 0.0037	13 0.0021
17.2998	1 0.2278	2 0.1968	3 0.1621	4 0.1283	5 0.0975	6 0.0697
17.2998	8 0.0323	9 0.0192	10 0.0117	11 0.0064	12 0.0038	13 0.0022
17.2998	1 0.2274	2 0.1965	3 0.1619	4 0.1283	5 0.0976	6 0.0698
17.2998	8 0.0325	9 0.0193	10 0.0118	11 0.0064	12 0.0038	13 0.0022
17.2998	1 0.2270	2 0.1962	3 0.1617	4 0.1282	5 0.0977	6 0.0699
17.2998	8 0.0327	9 0.0194	10 0.0119	11 0.0065	12 0.0039	13 0.0023
17.2998	1 0.2266	2 0.1959	3 0.1615	4 0.1282	5 0.0978	6 0.0700
17.2998	8 0.0329	9 0.0195	10 0.0120	11 0.0065	12 0.0039	13 0.0023
17.2998	1 0.2262	2 0.1957	3 0.1613	4 0.1281	5 0.0979	6 0.0701
17.2998	8 0.0331	9 0.0196	10 0.0121	11 0.0066	12 0.0040	13 0.0024
17.2998	1 0.2258	2 0.1954	3 0.1611	4 0.1281	5 0.0980	6 0.0702
17.2998	8 0.0333	9 0.0197	10 0.0122	11 0.0066	12 0.0040	13 0.0024
17.2998	1 0.2254	2 0.1951	3 0.1609	4 0.1280	5 0.0981	6 0.0703
17.2998	8 0.0335	9 0.0198	10 0.0123	11 0.0067	12 0.0041	13 0.0025
17.2998	1 0.2250	2 0.1948	3 0.1607	4 0.1280	5 0.0982	6 0.0704
17.2998	8 0.0337	9 0.0199	10 0.0124	11 0.0067	12 0.0041	13 0.0025
17.2998	1 0.2246	2 0.1945	3 0.1605	4 0.1279	5 0.0983	6 0.0705
17.2998	8 0.0339	9 0.0200	10 0.0125	11 0.0068	12 0.0042	13 0.0026
17.2998	1 0.2242	2 0.1942	3 0.1603	4 0.1279	5 0.0984	6 0.0706
17.2998	8 0.0341	9 0.0201	10 0.0126	11 0.0068	12 0.0042	13 0.0026
17.2998	1 0.2238	2 0.1939	3 0.1601	4 0.1278	5 0.0985	6 0.0707
17.2998	8 0.0343	9 0.0202	10 0.0127	11 0.0069	12 0.0043	13 0.0027
17.2998	1 0.2234	2 0.1936	3 0.1599	4 0.1278	5 0.0986	6 0.0708
17.2998	8 0.0345	9 0.0203	10 0.0128	11 0.0069	12 0.0043	13 0.0027
17.2998	1 0.2230	2 0.1933	3 0.1597	4 0.1277	5 0.0987	6 0.0709
17.2998	8 0.0347	9 0.0204	10 0.0129	11 0.0070	12 0.0044	13 0.0028
17.2998	1 0.2226	2 0.1930	3 0.1595	4 0.1277	5 0.0988	6 0.0710
17.2998	8 0.0349	9 0.0205	10 0.0130	11 0.0070	12 0.0044	13 0.0028
17.2998	1 0.2222	2 0.1927	3 0.1593	4 0.1276	5 0.0989	6 0.0711
17.2998	8 0.0351	9 0.0206	10 0.0131	11 0.0071	12 0.0045	13 0.0029
17.2998	1 0.2218	2 0.1924	3 0.1591	4 0.1276	5 0.0990	6 0.0712
17.2998	8 0.0353	9 0.0207	10 0.0132	11 0.0071	12 0.0045	13 0.0029
17.2998	1 0.2214	2 0.1921	3 0.1589	4 0.1275	5 0.0991	6 0.0713
17.2998	8 0.0355	9 0.0208	10 0.0133	11 0.0072	12 0.0046	13 0.0030
17.2998	1 0.2210	2 0.1918	3 0.1587	4 0.1275	5 0.0992	6 0.0714
17.2998	8 0.0357	9 0.0209	10 0.0134	11 0.0072	12 0.0046	13 0.0030
17.2998	1 0.2206	2 0.1915	3 0.1585	4 0.1274	5 0.0993	6 0.0715
17.2998	8 0.0359	9 0.0210	10 0.0135	11 0.0073	12 0.0047	13 0.0031
17.2998	1 0.2202	2 0.1912	3 0.1583	4 0.1274	5 0.0994	6 0.0716
17.2998	8 0.0361	9 0.0211	10 0.0136	11 0.0073	12 0.0047	13 0.0031
17.2998	1 0.2198	2 0.1909	3 0.1581	4 0.1273	5 0.0995	6 0.0717
17.2998	8 0.0363	9 0.0212	10 0.0137	11 0.0074	12 0.0048	13 0.0032
17.2998	1 0.2194	2 0.1906	3 0.1579	4 0.1273	5 0.0996	6 0.0718
17.2998	8 0.0365	9 0.0213	10 0.0138	11 0.0074	12 0.0048	13 0.0032
17.2998	1 0.2190	2 0.1903	3 0.1577	4 0.1272	5 0.0997	6 0.0719
17.2998	8 0.0367	9 0.0214	10 0.0139	11 0.0075	12 0.0049	13 0.0033
17.2998	1 0.2186	2 0.1900	3 0.1575	4 0.1272	5 0.0998	6 0.0720
17.2998	8 0.0369	9 0.0215	10 0.0140	11 0.0075	12 0.0049	13 0.0033
17.2998	1 0.2182	2 0.1897	3 0.1573	4 0.1271	5 0.0999	6 0.0721
17.2998	8 0.0371	9 0.0216	10 0.0141	11 0.0076	12 0.0050	13 0.0034
17.2998	1 0.2178	2 0.1894	3 0.1571	4 0.1271	5 0.1000	6 0.0722
17.2998	8 0.0373	9 0.0217	10 0.0142	11 0.0076	12 0.0050	13 0.0034
17.2998	1 0.2174	2 0.1891	3 0.1569	4 0.1270	5 0.1001	6 0.0723
17.2998	8 0.0375	9 0.0218	10 0.0143	11 0.0077	12 0.0051	13 0.0035
17.2998	1 0.2170	2 0.1888	3 0.1567	4 0.1270	5 0.1002	6 0.0724
17.2998	8 0.0377	9 0.0219	10 0.0144	11 0.0077	12 0.0051	13 0.0035
17.2998	1 0.2166	2 0.1885	3 0.1565	4 0.1269	5 0.1003	6 0.0725
17.2998	8 0.0379	9 0.0220	10 0.0145	11 0.0078	12 0.0052	13 0.0036
17.2998	1 0.2162	2 0.1882	3 0.1563	4 0.1269	5 0.1004	6 0.0726
17.2998	8 0.0381	9 0.0221	10 0.0146	11 0.0078	12 0.0052	13 0.0036
17.2998						

M/M/1 TRANSIENT QUEUEING PROBLEM BY BESSEL FUNCTIONS

STUART OLSON

INIT Q		NMAX	LAMBDA	PU	ALPHA					
0		21	0.4500	0.5000	C.9000					
19.7996	1 0.2305	2 0.1786	3 0.1637	4 0.1288	5 0.0966	6 0.0690	7 0.0470	8 0.0305	9 0.0188	10 0.0111
	9 0.0188	10 0.0111	11 0.0063	12 0.0034	13 0.0017	14 0.0009	15 0.0004	16 0.0002	17 0.0001	18 0.0000
19.8996	1 0.2301	2 0.1783	3 0.1635	4 0.1287	5 0.0966	6 0.0691	7 0.0471	8 0.0306	9 0.0190	10 0.0112
	9 0.0190	10 0.0112	11 0.0063	12 0.0034	13 0.0018	14 0.0009	15 0.0004	16 0.0002	17 0.0001	18 0.0000
19.9995	1 0.2297	2 0.1780	3 0.1633	4 0.1287	5 0.0967	6 0.0693	7 0.0473	8 0.0308	9 0.0191	10 0.0113
	9 0.0191	10 0.0113	11 0.0064	12 0.0035	13 0.0018	14 0.0009	15 0.0004	16 0.0002	17 0.0001	18 0.0000

P-VECTOR

APPENDIX C

RUNGE-KUTTA PROGRAM

The Chapman-Kolmogorov equations representing a queueing model are readily solvable using the Runge-Kutta method of solving differential equations. Experience has shown, however, that a technique of standardization of these equations and the form of the Runge-Kutta algorithm to solve them must be sought in order to obtain generality, convenience and efficiency. This section describes a programming package which will guide its user in obtaining a systematic time-dependent solution of the differential equations of queueing models.

It is assumed that the user of this programming package is familiar with the Chapman-Kolmogorov form of queueing equations and has a particular problem to be solved. Furthermore, it is assumed that the user is familiar with the FORTRAN-IV programming language.

The user should be acquainted enough with the Runge-Kutta algorithm to realize that it is a self-starting method which requires four evaluations of the differential equations per integration step h in the independent variable t . Preliminary evaluations are made twice with the independent variable at $t+h/2$, one at $t+h$ and then one final

evaluation with correct, updated solutions at $t+h$. In the program, these steps are referred to by 1, 2, 3 and 4. The variable k will also be used to stand for these numbers. The term self-starting above means that only values of the solution variables and differentials at time t are needed to obtain a new solution at time $t+h$.

With these preliminaries in mind, it is now possible to provide a concise summary of the basic scheme which will be presented. To obtain numerical solutions of differential equations using the Runge-Kutta method requires one to specify the equations in FORTRAN notation, specify the initial conditions, specify values for constants and parameters as required by the particular problem and to specify the range of the independent variable t and the step size h . These conditions must be met by the user for any problem.

Normally, in addition to specifying conditions relating directly to his problem, the user must implement the integration scheme, define some type of data structure for intermediate or working storage and construct an output format and control over print frequency.

The programming package presented here standardizes the method of defining the necessary conditions for any problem mentioned formerly and thus eliminates the necessity to plan, program and debug the latter steps relating to implementation detail. This is accomplished by requiring

the user to construct a subroutine named INIT with FORTRAN Entry points named PRIME, NEWVAL and WRAPUP. This subroutine will describe the user problem in standard form to the programming package.

Briefly, initial entry point INIT will be used by the user to specify initial conditions of the problem, to specify the range of the independent variable t and the step size. These data will be assigned to variables listed in a specified order in a COMMON statement. Entry point PRIME will serve to provide computed values of the differentials, given values of the solution variable during operation of the generalized Runge-Kutta subroutine. Through entry point NEWVAL, the package makes the solution variables and updated differentials available after each step h in t . Entry point WRAPUP is used after the problem is solved to allow the user to decide whether to start a new problem or terminate the computer run.

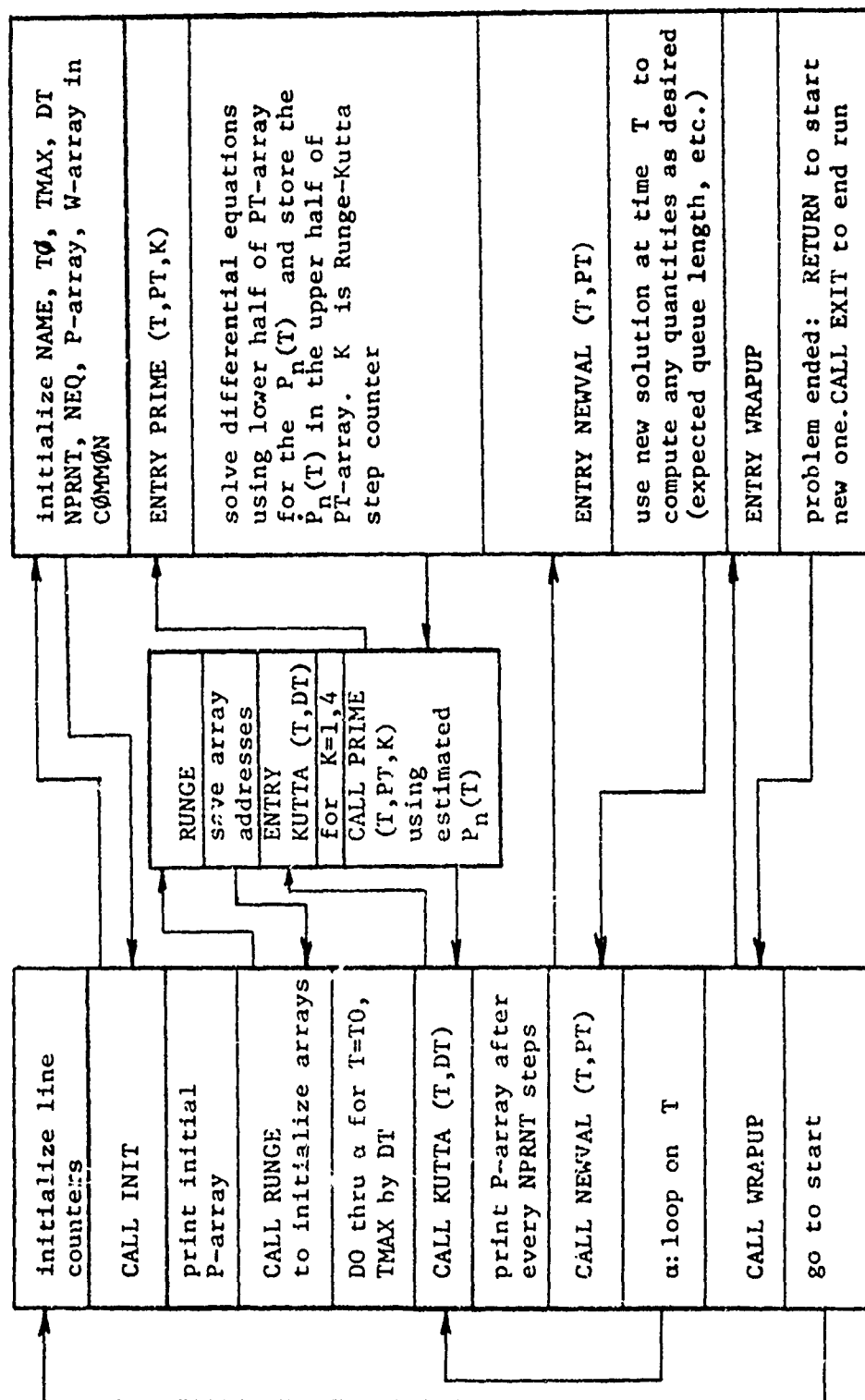
Table three Program Logic presented below shows how the user subroutine INIT interfaces with the programming package, which consists of a Main program and the Runge-kutta subroutine. It is not necessary for a user to understand the programming package itself, but only to understand the responsibilities required of the INIT subroutine at each of the points of interface.

Following the Program Logic, the next part of this section will consist of a detailed example of how the

equations for the M/M/1 queue were solved using this systematic programming package.

Table 3
Program Logic

USER-WRITTEN SUBROUTINE



Solution of the M/M/1 Queue

The equations for the M/M/1 queue are

$$\dot{P}_0(t) = -\lambda P_0(t) + \mu P_1(t) ,$$

$$\dot{P}_n(t) = \lambda P_{n-1}(t) - (\lambda + \mu) P_n(t) + \mu P_{n+1}(t) , \text{ for } 1 \leq n \leq N ,$$

$$\dot{P}_N(t) = \lambda P_{N-1}(t) - (\lambda + \mu) P_N(t)$$

where N is the maximum number of items in the system.

In this example $P_0(0)=1$ and $P_n(0)=0$ for $n=1;N$ are the initial conditions. The parameters are $\lambda=0.45$, $\mu=0.5$ and $N=20$. The variable t will run from 0 to 20 with Runge-Kutta step size of 0.1.

Although theoretically there is no reason to limit the maximum number in the system, in actual computation an upper bound must be chosen. One can either truncate negligible terms or modify the equations to take care of the finite waiting room, as was done above. The latter choice is recommended because it is then possible to utilize the auxiliary condition

$$\sum_{n=0}^{\infty} P_n(t) = 1$$

to compute a measure of solution error $\epsilon(t)$ from

$$\varepsilon(t) = 1 - \sum_{n=0}^N \hat{P}_n(t) .$$

The $\hat{P}_n(t)$ notation is used to distinguish between the notation used in the model and the actual computed values.

The usual practice is to assign initial conditions at $t=0$ and obtain the solution of the equations at selected positive values of t . The programming package will print the solution after every multiple of NPRNT Runge-Kutta steps chosen by the user.

Subroutine INIT

At this point, we have a statement of the necessary conditions of the sample problem and are ready to write a subroutine INIT for solving it. The following will be a step-by-step procedure for writing the FORTRAN subroutine INIT. Each FORTRAN statement will be followed by an explanation of it in relation to the problem to be solved. Statements marked by $\checkmark$ indicate the statement is always required by the programming package interface for any problem, except for the characters underscored, which are to be provided by the user as required by the problem. Statements not so indicated are user-specified to pertain to the problem at hand.

- $\checkmark$ 1) SUBROUTINE INIT
- $\checkmark$ 2) COMMON NAME(20), T0, TMAX, DT, NPRNT, NEQ,

P(21,2),W(63)

This statement is used as a common reference to locate data for the problem. Each array except W and each variable in CØMMØN must be assigned values as described below.

3) REAL LAMBDA,MU

Indicates λ and μ , respectively, in the equations will have these names and will be floating point reals.

✓ 4) READ (5,1,END=10) NAME,LAMBDA,MU,TMAX,DT

✓ 5) 1 FØRMAT (20A4/4F10.0)

Reads a problem identification card from the card reader and stores in array NAME. Reads the parameters λ and μ , the upper limit of time t for the equations TMAX and the step size for Runge-Kutta integration h, named DT in the program.

6) PRINT 2,NAME,LAMBDA,MU,TMAX,DT

7) 2 FØRMAT (1H1 20A4/1H0 9X6HLAMBDA 13X2HMU
11X4HTMAX 13X2HDT/1H0 4F15.4)

Prints out the input data.

✓ 8) TØ=0.0

Assign the initial value of time t, named T in the program. This could have also received its value in the READ statement as well. The problem is to be integrated from TØ to TMAX in steps of DT.

✓ 9) NPRNT=5

This is the number of DT steps in time t between printouts of the solution by the package Main program.

✓ 10) NEQ=21

The problem has 21 equations to be solved. Since FORTRAN does not allow a subscript of zero, all subscripts of the problem must be increased by one for the program.

11) $NM\emptyset = NEQ - 1$

12) $NP\emptyset = NEQ + 1$

13) $C\emptyset NS = LAMBDA + MU$

Constants used by this problem to control loops and prevent needless computation in later entry point sections of the subroutine.

✓ 14) $D\emptyset \ 3 \ I=2, NEQ$

✓ 15) 3 $P(I, 1) = 0.0$

✓ 16) $P(1, 1) = 1.0$

Initialization of the P-array. A later section titled Dimensioning the Problem explains the reason for an additional subscript for the P-array.

17) $LINE = 55$

18) $IPRNT = 0$

Auxiliary counters used by the problem program for additional output control.

✓ 19) RETURN

Returns to the Main program to complete problem initialization.

PRIME Entry Point

This is the entry point called from the Runge-Kutta subroutine during the four step estimation of the solution vector. The estimates are in an array named PT which is given to this entry point along with time T and the Runge-Kutta step number K. The user must use the values of PT(*,1) in the differential equations which are stored in PT(*,2). The * notation here stands for the subscripting pertaining to the users' problem, explained below in the section titled Dimensioning the Problem.

✓ 20) ENTRY PRIME(T,PT,K)

✓ 21) DIMENSION PT(21,2)

The PT-array must be dimensioned exactly as the P-array was in the COMMON statement above.

✓ 22) PT(1,2)=RHS(MU*PT(2,1)-LAMBDA*PT(1,1))

First user problem equation solved and stored in upper half of P-array. Enclosing the expression in parentheses as an argument to the RHS function prevents machine underflow interrupts.

✓ 23) DO 4 I=2,NMØ

Loop counter for the general equation of user problem.

✓ 24) 4 PT(I,2)=RHS(LAMBDA*PT(I-1,1)-CONS*PT(I,1)
+MU*PT(I+1,1))

The general equation of the user problem is computed and stored in the upper half of the PT-array.

✓ 25) PT(NEQ,2)=RHS(LAMBDA*PT(NMØ,1)-CONS*PT(NEQ,1))

The final equation for the user problem is computed and stored in the upper half of the PT-array.

✓ 26) RETURN

Return control to the Runge-Kutta subroutine.

NEWVAL Entry Point

This entry point is called after each step in the solution, i.e., after each time that a new solution vector is computed and $T \leftarrow T + DT$. The user is given the new value of time T and the solution vector and updated derivatives in the PT-array.

✓ 27) ENTRY NEWVAL(T,PT)

In this problem it is desired to compute and print the solution error after every NPRNT steps of DT in T and label the output at the top of each page.

28) CALL CØNTRL(IPRNT,NPRNT,&11)

CØNTRL is part of the programming package. Using the free counter variable IPRNT and counter limit NPRNT, it returns to statement 11 whenever IPRNT is less than NPRNT after incrementing IPRNT by one. It returns to the next statement if IPRNT equals NPRNT after setting IPRNT to zero.

29) CALL CØNTRL(LINE,55,&5)

This counter switch, used for page control, is similar to the above. It executes the next statement after every 55 passes.

30) PRINT 7,NAME

Print column headings at top of each page, that is, after every 55 passes through this section.

```
31)      7  FØRMAT (1H1 20A4/1H0 14X 1HT 6X
           14HERRØR, PERCENT)
```

```
32)      5  SUM=0.0
```

Initialize for computing the solution error.

```
33)      DØ 8 I=1,NEQ
```

Loop for computing sum of solution vector.

```
34)      8  SUM=SUM+PT(NPØ-I,1)
```

```
35)      ERRØR=100.0*(SUM-1.0)
```

Compute percent error of solution.

```
36)      PRINT 9,T,ERRØR
```

Print current time T and percent error.

```
37)      9  FØRMAT(1H F15.4,F20.6)
```

```
✓ 38)    11  RETURN
```

Any problem statistics such as the expected number in the system would be computed in entry point NEWVAL by the user as required by his particular problem.

WRAPUP Entry Point

This entry point is called from the Main program after the problem has been solved, that is, when time T reaches TMAX. If the user executes a RETURN, the Main program calls INIT, thus providing the user the opportunity to initialize a new problem. A CALL EXIT statement, written instead, terminates the run.

```
✓ 39)      ENTRY WRAPUP  
      40)      RETURN
```

In this example, the RETURN causes a call to INIT. The READ statement is executed to get new problem parameters. If no more data is in the reader, a branch is made to statement 10 to terminate the run.

```
      41)      10  CALL EXIT  
✓ 42)      END
```

The FORTRAN listing of this program and sample output are included next. The section following the listings will discuss in detail the problem of storage of the probability vectors in the P-array.

Reproduced from
best available copy.

PAGE 0001

15/23/42

DATE = 72125

INIT

FORTRAN IV G LEVEL 20

0001 SUBROUTINE INIT

C
C*** SIMPLE M/M/1 QUEUE.C
C*** CUMPHO1 NAME(2),T0,TMAX,DT,NPRINT,NEQ,P(21,2),M(63)C
C*** THE SYSTEM HAS 21 STATES AND IS OF ORDER 1, THEREFORE THE P-MATRIX
C*** IS DIMENSIONED 21 BY 2. THE M ARRAY IS ALWAYS 1.5 TIMES THE SIZE
C*** OF P, THEREFORE IT IS DIMENSIONED FOR 21*2*1.5=63.C
C*** READ LAMBDA,MUC
C*** READ (5,1,END=10) NAME,LAMBDA,MU,TMAX,DTC
C*** 1 FORMAT (2E4,4F10,0)C
C*** 2 FORMAT (1H12,4F10,9X6H LAMBDA 13X2H MU 11X4H TMAX 13X2H DT /C
C*** 100,4,15,4)C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0C
C*** 100,0,0

```

FCRTRAM IV G LEVEL 2C          DATE = 72125          PAGE 00.2
C
C*** COMPUTE THE ERROR AND PRINT AFTER EVERY NPANT STEPS.
C
C028 CALL CONTRL(IPRINT,NPANT,E11)
C029 CALL CONTRL(LINE,55,65)
C030 PRINT 7,NAME
C031 7 FORMAT (1H12LA4/1H0 14XINT 6X14ERROR, PERCENT)
C032 5 SUM=0.0
C033 DO 6 I=1,NEQ
C034 6 SUM=SUM+PTINPO-I,1)
C035 8R204=100.0*(SUM-1.0)
C036 PRINT 9,I,ERRMGR
C037 9 FORMAT (1H F15.4,F20.6)
C038 11 RETURN
C
C039 ENTRY WRAPUP
C040 RETURN
C041 10 CALL EXIT
C042 END

```

Reproduced from
best available copy.



```

FORTRAN IV G LEVEL 20          MAIN          DATE = 72103          18/11/36          PAGE 0001

C*** MAIN CONTROL MODULE FOR RUN. --KUTTA QUEUEING EQUATION SOLVER.
C*** STUART OLSON -- APRIL 1972 --REV. OF IONA -- 56-299
C
0001 COMMON NAME(20),T0,TMAX,DT,NPRINT,NEQ,P(11)
0002 DATA EPS/5.0E-5/
0003 INTEGER *2 IP(2),* *O*,IH
0004
0005 CALL INIT
0006 *STAR=1
0007 DO 2 I=1,NEQ
0008 IF (P(I)-GT.0.0) NSTAR=I
0009 2 CONTINUE
0010 NPO=NEQ+1
T=T0
C
C*** SOLVE DIFF-DIFF EQNS. AT INITIAL CONDITIONS.
C
0011 CALL PRIME(T,P,1)
C
C*** MAKE WORKING-STORAGE ASSIGNMENTS TO RUNGE-KUTTA SUBROUTINE.
C
0012 CALL RUNGE(P(1),P(2*NEQ+1),P(4*NEQ+1),NEQ,1,2)
C
C*** PRINT-OUT INITIAL P-VECTOR.
C
WRITE (P,3) NAME
3 FORMAT (1H12CA4/IHO 8XINT3XRRHP-VECTOR)
IH=IP(2)
IF (N.LE.10) IH=IP(1)
WRITE (P,4) IH,T,1,P(1),1,1,N
4 FORMAT (A1,F9.2,10I4,F7.4)
IF (NSTAR.LE.10) GO TO 5
N=N+1
WRITE (P,6) (1,P(I),1,N,NSTAR)
6 FORMAT (1H 9X,14,F7.4,14,F7.4,14,F7.4,14,F7.4,14,F7.4,14,F7.4,
* 14,F7.4,14,F7.4,14,F7.4,14,F7.4,14,F7.4)
5 LINE=-52*(NSTAR+9)/10
IP=NT=C
7 NT=(TMAX-T0)/DT+0.5
C
C*** START INTEGRATION LOOP. STEP FROM T0 TO TMAX IN STEPS OF DT.
C
DO 8 IT=1,NT
C
C*** SOLVE DIFF-DIFF EQNS. FOR T=TDOT.
C
CALL KUTTA(T,DT)
C
0028

```



```

0031      DO 7 J=1,NGRID
0032      DO 7 I=1,NEQ
0033      7 V(I,J)=V(I,J)+DT6*O(I,J)
0034      T=TW
0035      CALL PRIME(T,V,4)
0036      RETURN
0037      END

```

RUNGE
 DATE = 72103
 18/11/36
 PAGE 0002

FORTRAN IV G LEVEL 2C

```

FORTRAN IV G LEVEL 20          DATE = 72103          18/11/36          PAGE 0001

0001      FUNCTION RHS(P)
C
C*** THIS FUNCTION IS REQUIRED TO PREVENT EXPONENT UNDERFLOW BY
C*** ARBITRARILY CUTTING OFF SMALL ABSOLUTE VALUES BEFORE THEY
C*** CAN CAUSE A MACHINE UNDERFLOW. (ON 360/65 UNDERFLOW OCCURS WHEN
C*** A NUMBER IS LESS THAN 16**66 AFTER A COMPUTATION.)
C
      RVAL=EPS/1.0E-70/
      IF (ABS(P).LT.EPS) GO TO 1
      RHS=P
      RETURN
      1 RHS=0.0
      RETURN
      END
0002
0003
0004
0005
0006
0007
0008

```

```

FORTRAN IV G LEVEL 20          CONTRL          DATE = 72103          18/11/36          PAGE 0001
0001          C          SUBROUTINE CONTRL(I,N,*)
          C          C*** THIS SUBROUTINE HAS TWO EXIT POINTS -- ONE FOR WHEN I<N AND
          C          C*** ONE FOR I=N, IN WHICH CASE I IS ALSO SET TO ZERO.
          C          I=I+1
          C          IF (I.LT.N) RETURN I
          C          I=0
          C          RETURN
          C          END
0002
0003
0004
0005
0006

```

SINGLE-CHANNEL, SINGLE-SERVER QUEUE TO TEST EFFICIENCY OF R-K METHOD S-OLSDM

LAMBDA	MU	TMAX	DT
0.4500	0.5000	20.0000	0.1000

SINGLE-CHANNEL, SINGLE-SERVER QUEUE TO TEST EFFICIENCY OF R-K METHOD S. OLSON

T	ERROR, PERCENT
0.5000	-0.260012
1.0000	-0.000036
1.5000	-0.000072
2.0000	-0.000125
2.5000	-0.000167
3.0000	-0.000221
3.5000	-0.000268
4.0000	-0.000334
4.5000	-0.000387
5.0000	-0.000435
5.5000	-0.000531
6.0000	-0.000543
6.5000	-0.000608
7.0000	-0.000668
7.5000	-0.000721
8.0000	-0.000741
8.5000	-0.000846
9.0000	-0.000912
9.5000	-0.000989
9.9999	-0.001061
10.4999	-0.001132
10.9999	-0.001192
11.4999	-0.001276
11.9999	-0.001335
12.4999	-0.001413
12.9999	-0.001502
13.4999	-0.001597
13.9999	-0.001675
14.4999	-0.001746
14.9999	-0.001895
15.4999	-0.002027
15.9999	-0.002170
16.4999	-0.002349
16.9999	-0.002551
17.4999	-0.002913
17.9999	-0.003117
18.4999	-0.003475
18.9999	-0.003910
19.4999	-0.004423
19.9999	-0.005013

SINGLE-CHANNEL, SINGLE-SERVER DEFECT TO TEST EFFICIENCY OF R-K METHOD S. JILSON

T	P-VECTOR	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000
---	----------	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------

SINGLE-CHANNEL, SINGLE-SERVER QUEUE TO TEST EFFICIENCY OF R-K METHOD S. OLSON

T	P-VECTOR	1	2	3	4	5	6	7	8	9	10
16.00	1 C-2464	2 C-2160	3 C-1697	4 C-1798	5 C-0921	6 C-0620	7 C-0394	8 C-0235	9 C-0133	10 C-0071	
	11 C-0036	12 C-0017	13 C-0006	14 C-0003	15 C-0001	16 C-0001					
16.50	1 C-2464	2 C-2160	3 C-1697	4 C-1798	5 C-0921	6 C-0620	7 C-0405	8 C-0245	9 C-0140	10 C-0076	
	11 C-0039	12 C-0019	13 C-0009	14 C-0004	15 C-0002	16 C-0001					
17.00	1 C-2464	2 C-2160	3 C-1697	4 C-1798	5 C-0935	6 C-0641	7 C-0416	8 C-0255	9 C-0148	10 C-0081	
	11 C-0042	12 C-0021	13 C-0010	14 C-0004	15 C-0002	16 C-0001					
17.50	1 C-2464	2 C-2160	3 C-1697	4 C-1798	5 C-0940	6 C-0650	7 C-0426	8 C-0264	9 C-0155	10 C-0087	
	11 C-0046	12 C-0023	13 C-0011	14 C-0005	15 C-0002	16 C-0001					
18.00	1 C-2464	2 C-2160	3 C-1698	4 C-1798	5 C-0945	6 C-0659	7 C-0436	8 C-0273	9 C-0162	10 C-0092	
	11 C-0049	12 C-0025	13 C-0012	14 C-0006	15 C-0003	16 C-0001					
18.50	1 C-2464	2 C-2160	3 C-1698	4 C-1798	5 C-0950	6 C-0668	7 C-0445	8 C-0282	9 C-0170	10 C-0097	
	11 C-0053	12 C-0029	13 C-0014	14 C-0006	15 C-0003	16 C-0001	17 C-0001				
19.00	1 C-2464	2 C-2160	3 C-1699	4 C-1799	5 C-0955	6 C-0675	7 C-0454	8 C-0290	9 C-0177	10 C-0102	
	11 C-0057	12 C-0030	13 C-0015	14 C-0007	15 C-0003	16 C-0001	17 C-0001				
19.50	1 C-2464	2 C-2160	3 C-1699	4 C-1799	5 C-0958	6 C-0683	7 C-0463	8 C-0299	9 C-0184	10 C-0108	
	11 C-0060	12 C-0032	13 C-0016	14 C-0008	15 C-0004	16 C-0002	17 C-0001				
20.00	1 C-2464	2 C-2160	3 C-1699	4 C-1799	5 C-0962	6 C-0690	7 C-0471	8 C-0307	9 C-0190	10 C-0113	
	11 C-0064	12 C-0035	13 C-0018	14 C-0009	15 C-0004	16 C-0002	17 C-0001				

Dimensioning the Problem

The P-array in COMMON storage contains the solution vector and the derivatives. The dimension of the array, denoted by * below, is constructed to correspond to the possibly several subscripts of the user problem according to normal FORTRAN conventions. For each probability there is a corresponding derivative with the same subscript. By appending an additional subscript on the P-array with the value of two, two identical problem areas are set up. The first, that is, the one with the appended subscript value of one, is for the solution vector probabilities. The second is for the corresponding derivatives. The compiler assigns the storage in ascending order so that the first half is the lower part and the second is the upper part. Thus P(*,2) in the COMMON statement sets up two sequential arrays each of dimension *, where * here stands for the dimension of the user problem as discussed below in more detail. The user must assign NEQ a value equal to the product of the sizes of each component of * above. The main program is then able to locate the derivative locations by simply adding NEQ to the value of the subscript of the solution vector expression during the computations.

The Runge-Kutta subroutine requires working storage for intermediate results. This depends on the size of the user problem and thus must be assigned by the user in the COMMON statement as the W-array. The W-array will always be

dimensioned as exactly one-and-one-half times the size of the $P(*,2)$ -array. An example will be given to help clarify this point.

Suppose a problem has equations

$$\dot{P}_{ijk}(t) = f(t, P_{ijk}(t))$$

where i, j and k are indices used to enumerate the state of the system. Suppose further for this problem that

$$1 \leq i \leq 21$$

$$1 \leq j \leq 11$$

$$1 \leq k \leq 6.$$

In the notation used above, the problem dimension * is (21,11,6). The P-array, however, must be dimensioned for this problem with an appended subscript of value two. Therefore, in order to make space available for corresponding derivative values, write $P(21,11,6,2)$. Now the W-array must be one-and-one-half the size of the P-array. Write $W(4158)$, which is the product $(21)(11)(6)(2)(1.5)$.

Occasionally, a problem may be written which has more subscripts than the compiler will allow. (For the IBM 360 FORTRAN-IV compiler the maximum is seven.) Suppose the problem appears as

$$P_{i_1, i_2, i_3, \dots, i_p}(t) .$$

There are p subscripts. Also suppose these run from one to $m_1, m_2, m_3, \dots, m_p$, respectively. The p subscripts must be replaced by a lower number of subscripts for the compiler. Say one is used, denoted by j . Given a set of p indices in the program, the following algorithm should be used to compute j which will be then used as the subscript of the P -array.

Initially set $j = i_p - 1$. Then compute

$$j + j m_{p-k} + i_{p-k} - 1 \quad \text{for } k=1 \text{ to } p-1,$$

and then compute

$$j + j + 1.$$

To illustrate, suppose the four subscripts of the former example could not be dimensioned as $P(21, 11, 6, 2)$ as was done because of compiler restrictions. To use equivalent single-subscripting first dimension the P -array for the product of the subscripts, $P(2772)$. Here $m_1 = 21$, $m_2 = 11$, $m_3 = 6$, $m_4 = 2$ and $p = 4$. To locate the correct subscript for the P -array for some particular choice of subscripts, say $(i_1, i_2, i_3, i_4) = (2, 4, 5, 1)$, use the above algorithm. Set $j = i_4 - 1 = 1 - 1 = 0$, then for $k = 1$ to 3;

$$j+jm_1+i_1-1=(0)(6)+5-1=4,$$

$$j+jm_2+i_2-1=(4)(11)+4-1=47,$$

$$j+jm_1+i_1-1=(47)(21)+2-1=988,$$

and finally

$$j+j+1 = 988+1 = 989.$$

Thus P(989) corresponds to P(2,4,5,1).

Program Output

The programming package treats the P-array as if it were singly-subscripted. During a problem solution, the P-array will be printed after every NPRNT integration steps on FORTRAN unit 8. The output will consist of the problem NAME at the top of the page with the left column labeled for time T. After each value of T printed, the P-array will be printed immediately to the right. Values of the array which are less than 5×10^{-5} will not be printed if they follow larger values. If the single-subscript output form of the P-array is not desired by the user, define FORTRAN data set 8 as DUMMY in the operand field of the DD control card for FT08F001 as shown below in the Job Control Cards section and

Job Control Cards

The job control language (JCL) statements for running on the IBM 360/65 computer at the University of Iowa are now presented.

```
//QUEUEING JOB (XXXXXXXX),'STU QUEUES'
```

```
// EXEC PDQFOR,PARM='LIST/PRINT,MAP'
```

```
//SOURCE DD *
```

(Place source deck of programming package and subroutine
INIT here.)

```
/*
```

```
//DATA DD *
```

(Place any data to be read by subroutine INIT from
unit 5 here.)

```
/*
```

```
//FT08F001 DD SYSOUT=A,DCB=RECFM=UA
```

(To get P-array output from Main program.)

```
//FT08F001 DD DUMMY
```

(To ignore output from Main program.)

```
//SYSABEND DD SYSOUT=A
```

(Include if a dump is desired if abnormal termination
such as 0C4 or 0C5 occurs.)