THE DEVELOPMENT OF A PC BASED SOFTWARE TO
SOLVE M/M/1 AND M/M/S QUEUEING SYSTEMS BY
USING A NUMERICAL INTEGRATION TECHNIQUE/

A Thesis Presented to
The Faculty of the Russ College of
Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirement for the Degree
Master of Science

by
Jinchun Ho
November 1994
<table>
<thead>
<tr>
<th>PAGE</th>
<th>IV.3 Runge-Kutta Method</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IV.4 The Solution of Using Runge-Kutta Method in the Simulation Model</td>
<td>21</td>
</tr>
<tr>
<td>V.</td>
<td>RESULTS</td>
<td>24</td>
</tr>
<tr>
<td>VI.</td>
<td>DISCUSSION AND CONCLUSIONS</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>REFERENCES</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>APPENDIX</td>
<td>39</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>Description</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.1</td>
<td>Queueing Notations</td>
<td>4</td>
</tr>
<tr>
<td>I.2</td>
<td>Postulate of Birth and Death Process</td>
<td>5</td>
</tr>
<tr>
<td>III.1</td>
<td>Selection of The Best Alternatives</td>
<td>11</td>
</tr>
<tr>
<td>IV.1</td>
<td>Form Listing</td>
<td>13</td>
</tr>
<tr>
<td>V.1</td>
<td>Examples For Testing The Software</td>
<td>24</td>
</tr>
<tr>
<td>V.2</td>
<td>Results Hard Copy For Example 1</td>
<td>26</td>
</tr>
<tr>
<td>V.3</td>
<td>Results Hard Copy For Example 2</td>
<td>29</td>
</tr>
<tr>
<td>V.4</td>
<td>Results Hard Copy For Example 3</td>
<td>32</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

FIGURE

PAGE

IV.2.1 Programming Flow Chart ..................... 16
IV.2.2 Programming Flow Chart (Cont.).............. 17
IV.2.3 Programming Flow Chart (Cont.).............. 18
IV.2.4 Programming Flow Chart (Cont.).............. 19
V.1 Information Hard Copy For Example 1...... 25
V.2 Result Charts For Example 1................. 27
V.3 Information Hard Copy For Example 2...... 28
V.4 Result Charts For Example 2................. 30
V.5 Information Hard Copy For Example 3...... 31
V.6 Result Charts For Example 3................. 33
V.7 Result Charts For Examples................. 34
CHAPTER I INTRODUCTION

For years, queueing has been an interesting field of study in operations research and a variety of other disciplines. Theories and approaches have also been developed. However, problems have never been handled easily until, in recent years, the power of digital computers began booming continuously. They became vital tools, especially the PC (Personal Computer), for much scientific and engineering research.

Simulation is, in fact, the most widely used solution for queueing problems. It has been defined by Robert E. Shannon (1975) as "the process of designing a computerized model of a system (or Process) and conducting experiments with this model for the purpose either of understanding the behavior of the system or evaluating various strategies for the operation of the system." Though, the literature gives many definitions which seem to encompass the more important aspects of this problem-solving process.

There are basically two types of systems relating to simulation, continuous and discrete. The terms continuous and discrete applied to a system refer to the nature or behavior of changes with respect to time in the system state. A system whose changes in state occur continuously over time are continuous systems; systems whose changes occur in finite quanta or jumps are discrete systems. Some of the system state variables may vary continuously in response to events while others may vary discretely. Such systems can validly be labeled hybrid systems.

The discrete systems have been studied extensively and a variety of computer languages, programs and software have been developed. GPSS, SIMSCRIPT II.5 and GASP IV, etc. are beneficial to those who study the discrete oriented problems.

Although many theories have been developed for continuous systems, more efforts are still needed in terms of computer software which implement these theories. The difficulty lies in the complexity of the system, especially in the transient situation solving period. It requires the capability to solve a set of differential difference equations spontaneously.
The solutions of a queueing system usually can be classified into two categories, the steady state or transient solution. In steady state solution, the answer can be easily obtained through a straight mathematical calculation or by a transformation process. The straight mathematical approach, however, does not seem infeasible in the transient period because complex mathematical calculations are involved, although, this approach was used by Ledermann, W. and Reuter, G.E.H. in 1954. A paper published by Ronald L. Gue (1966) addresses the use of signal flow graphs in connection with an analog computer to obtain transient solutions for finite queueing systems with constant parameters. Helmut T. Zwahlen (1968) generated extensive equations and time dependent solutions using CSMP simulation and mathematical approaches in his thesis.

In recent years, numerical methods were developed to solve many sophisticated mathematical models by numerical approximation. The solution of a continuous queueing system became feasible in this approach and quite useful in the real world today.

This study focuses on the continuous systems and develops a user friendly interface software package which can be used in an operations research class.

I.1 REVIEW OF LITERATURE

I.1.1 Original Studies

The field of queueing was first studied by A.K. Erlang in early 1900s. His publication "Use of Waiting-line Theory in the Danish Telephone System" was published in 1908. He was also responsible for the notion of stationary equilibrium, for the introduction of the so called balance-of-state equations, and for the first consideration of the optimization of a queueing system. Work on the application of the telephone continued after Erlang. In 1927, Molina published his "Application of the Theory of Probability to Telephone Trunking Problems" which was followed by Thornton Fry's "Probability and It's Engineering Uses," which explained much of Erlang's earlier work. In the early 1930s, Felix Pollaczek did some further pioneering work on Poisson input, arbitrary output, and single-, and multiple-channels problems. Additional work was done at the time in Russia by Kolmogorov and Khintchine, in France by Crommelin, and in Sweden by Palm. The early work in queueing theory picked up
momentum rather slowly but the trend began to change in the 1950s and, of late, there has been much work in the area. The application of queueing theory was limited until the mid 1940s when the field of operations research brought the queueing process to the attention of many scientists. An estimated two thousand papers were published during this period. Since then, the applications range from fairly simple waiting line situations in barber shops or at ticket counters to sophisticated decision making in industrial or commercial operations. The basic nature of queueing situations has been discussed by several authors including Bhatia and Garg (1963), Hiller (1964), Lee (1966), Patterson (1964) and Lajos (1962).

I.1.2 The Queueing Process

A queueing system can be described as customers arriving for service, waiting for service if it is not immediate and, if having waited for service, leaving the system after being served. There are six basic characteristics of a queueing process as follows:

1. Arrival pattern of customers.
2. Service pattern of servers.
3. Queue discipline.
4. System capacity.
5. Number of service channels.
6. Number of service stages.

In most cases, these six basic characteristics provide an adequate description of a queueing system.

I.1.3 Notation

The standard notation throughout the queueing literature was introduced by D.G. Kendall (1951). A queueing process is described by a series of symbols and slashes such as A/B/X/Y/Z, where A indicates in some way the interarrival time distribution, B the service pattern as described by the probability distribution for service time, X the number of parallel service channels, Y the restriction on system capacity, and Z the queueing discipline. Some standard symbols for these characteristics are presented in the following table:
Table I.1 Queueing Notation

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Symbol</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interarrival-time distribution (A)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>Exponential</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Deterministic</td>
</tr>
<tr>
<td></td>
<td>$E_k$</td>
<td>Erlang type $k$ ($k=1,2,...$)</td>
</tr>
<tr>
<td></td>
<td>$H_k$</td>
<td>Hyperexponential type $k$</td>
</tr>
<tr>
<td></td>
<td>PH</td>
<td>Phase type</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>General</td>
</tr>
<tr>
<td>Service-time distribution (B)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>Exponential</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Deterministic</td>
</tr>
<tr>
<td></td>
<td>$E_k$</td>
<td>Erlang type $k$</td>
</tr>
<tr>
<td></td>
<td>$H_k$</td>
<td>Hyperexponential type $k$</td>
</tr>
<tr>
<td></td>
<td>PH</td>
<td>Phase type</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>General</td>
</tr>
<tr>
<td>Number of parallel servers (X)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$1,2,\ldots\infty$</td>
<td></td>
</tr>
<tr>
<td>Restriction on system capacity (Y)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$1,2,\ldots\infty$</td>
<td></td>
</tr>
<tr>
<td>Queueing discipline (Z)</td>
<td>FCFS</td>
<td>First come, first served</td>
</tr>
<tr>
<td></td>
<td>LCFS</td>
<td>Last come, first served</td>
</tr>
<tr>
<td></td>
<td>RSS</td>
<td>Random selection for service</td>
</tr>
<tr>
<td></td>
<td>PR</td>
<td>Priority</td>
</tr>
<tr>
<td></td>
<td>GD</td>
<td>General discipline</td>
</tr>
</tbody>
</table>

I.1.4 The Focusing of Study

The most common stochastic queueing models are described by the Markov process where the arrival process is a poisson process and service times follow a negative exponential distribution. Although there are other arrival and service processes, this study focuses on only those systems that may be described or simulated by a Markov process. The notation associated with these systems will be M/M/No. of server/max.no. in system/FCFS.

I.1.5 The Mathematical Model

The elementary of a queueing system is according to the birth-death process. The term "birth" refers to the arrival
of a new calling unit into the system and "death" refers to the departure of a served unit. In Hillier and Lieberman's publication (1967), a formulation was established.

For \( n > 0 \), the system could reach the state \( E_n \) at time \((t + \Delta t)\) from its state at time \( t \) in one of the four mutually exclusive ways described in the follow table:

Table I.2 Postulate of Birth and Death Process

<table>
<thead>
<tr>
<th>State at ( t )</th>
<th>Even From ( t ) to ((t + \Delta t))</th>
<th>Unconditional Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>( E_{n-1} )</td>
<td>One birth</td>
<td>( P_{n-1}(t)[\lambda_{n-1}\Delta t + 0\Delta t] )</td>
</tr>
<tr>
<td>( E_{n+1} )</td>
<td>One death</td>
<td>( P_{n+1}(t)[\lambda_{n+1}\Delta t + 0\Delta t] )</td>
</tr>
<tr>
<td>( ? )</td>
<td>Multiple events</td>
<td>( 0\Delta t )</td>
</tr>
<tr>
<td>( E_n )</td>
<td>None</td>
<td>( P_n(t)[1-\lambda_n\Delta t - \mu_n\Delta t + 0\Delta t] )</td>
</tr>
</tbody>
</table>

The above postulate follows that

\[
P_n(t + \Delta t) = P_{n-1}(t)[\lambda_{n-1}\Delta t + 0\Delta t] + P_{n+1}(t)[\lambda_{n+1}\Delta t + 0\Delta t] + 0\Delta t + P_n(t)[1-\lambda_n\Delta t - \mu_n\Delta t + 0\Delta t]
\]

Combining \( 0\Delta t \) terms,

\[
P_n(t + \Delta t) = P_{n-1}(t)\lambda_{n-1}\Delta t + P_{n+1}(t)\mu_{n+1}\Delta t + P_n(t)[1-\lambda_n\Delta t - \mu_n\Delta t] + 0(\Delta t)
\]

Subtracting \( P_n(t) \) from both sides and dividing through by \( \Delta t \) yields

\[
(P_n(t + \Delta t) - P_n(t))/\Delta t = \lambda_{n-1}P_{n-1}(t) + \mu_{n+1}P_{n+1}(t) - (\lambda_n + \mu_n)P_n(t) + 0(\Delta t)/\Delta t
\]

Since the equation holds for all positive value of \( \Delta t \),

\[
\lim_{\Delta t \to 0} \{(P_n(t + \Delta t) - P_n(t))/\Delta t \} = \lim_{\Delta t \to 0}\{(\lambda_{n-1}P_{n-1}(t) + \mu_{n+1}P_{n+1}(t) - (\lambda_n + \mu_n)P_n(t) + 0(\Delta t)/\Delta t \}
\]
By definition, the left side is the derivative and the last term on right side equals zero. Therefore,

\[ \frac{dP_n(t)}{dt} = \lambda_{n-1} P_{n-1}(t) + \mu_{n+1} P_{n+1}(t) - (\lambda_n + \mu_n) P_n(t) \quad \text{for } n > 0 \]

When \( n=0 \) set, \( \lambda_{n-1} = 0 \) and \( \mu_0=0 \), so that

\[ \frac{dP_0(t)}{dt} = \mu_1 P_1(t) - \lambda_0 P_0(t) \]

This provides a set \( (n = 0, 1, 2, 3, \ldots) \) of differential equations which if they could be solved, would provide the value of \( P_n(t) \).

1.2 OBJECTIVES OF STUDY.

The objective of this study is to design a PC based computer software package which will be useful in an operations research class and will allow students to obtain transient solutions to a selected class of queueing problems.
CHAPTER II SYSTEM REQUIREMENTS

In order to develop a computer system which accomplishes the objectives of this study, some requirements were set along with the development.

II.1 Hardware Requirements

In order to be able to used for teaching in the classroom, the software package must be a PC based software. A 80486 CPU is recommended since the Runge-Kutta method requires a large amount of numerical integration. In general, the requirements of the hardware are as follows:

1. Personal Computer with 486 CPU.
2. 4 MB or higher of RAM.
3. 33 MHz speed.
4. Hard disk.
5. Mouse.
7. Printer.

II.2 Software Requirements

Because of its popularity and it is a user friendly interface, Microsoft windows 3.1 operation system is required in order to let students or users learn quickly.

II.3 Input Requirements

Before the user begins to use the package, he/she must realize that the system can be run under the following assumptions:

1. Arrival rate is poisson distributed.
2. Service rate is negative exponential distributed.
3. If maximum number of the system is full, the coming arrival will be lost.
4. Infinite arriving resource.

The requirements for inputs are as follows,

1. Maximum number in the system must be an integer.
2. Number of servers must be an integer.

3. The rest of the inputs can be either integers or real numbers.

4. The default system type is M/M/1. It can be changed to M/M/s.

5. The default integration interval is 0.05 hr. and print every 10 integrations. They can be changed.

6. All input data can be saved in a file and can be retrieved from a file.

II.4 Output Requirements

The requirement of output including the screen and hard copy.

The results includes the following:

1. The illustration of the queueing model.

2. The Transition Rate Matrix.

3. The Differential-Difference equations.

4. The probabilities against time table.

5. Expected values in the system.

6. Expected values in the queue.

7. Tracking of lost when system is full.

CHAPTER III

DESIGN ALTERNATIVES AND SELECTION OF BEST ALTERNATIVE

III.1 Hardware Alternatives

There are two alternatives to developing the software package in this category, one is executing under a mainframe system and the other can be run by a personal computer.

The advantage of using a mainframe system is its huge memory base which is beneficial to running speed if an extra large queueing model is involved. However, the software lacks portability from one mainframe system to another or to a PC. In addition, there is no user friendly interface such as Microsoft windows available for mainframe.

On the other hand, the advantage of using a PC is the software's portability and the availability of user friendly interfaces such as the Microsoft windows operating system. The only disadvantage of using a PC is that it requires a longer time period to run a large scale model.

III.2 Software Alternatives

If other than mainframe, the software alternatives are basically of two types. The first one is using the DOS system on a PC and the other is utilizing Microsoft windows environment on a PC.

Although most of the PC's operating system is DOS and Microsoft windows is actually run under DOS, Microsoft windows has much more powerful features such as more efficient memory management functions.

Most importantly, Microsoft windows has a very nice interface that users can maneuver through every application with minimum hardship memorizing every command or function key.
III.3 Selection of the Best Alternatives

In order to conclude the best alternatives, a ranking and weighting analysis has been done to compare some design alternatives against some requirements. First, a list of design requirements and a list of alternatives related to these requirements are as follows:

Requirements
1. Portability.
2. User friendly.
3. Easy to program.

Alternatives
1. Use mainframe.
2. Use PC.
3. Use DOS.
4. Use Microsoft Windows.

Secondly, rank scaled from zero to five was assigned to each alternative in a requirement category. A percentage weight that represented the importance was also assigned to the alternatives. The product of rank and scale represents the desirability/preferability of each alternative. The result is shown in table III.1
Table III.1 Selection of The Best Alternatives

<table>
<thead>
<tr>
<th></th>
<th>Req.1</th>
<th></th>
<th></th>
<th>Req.2</th>
<th></th>
<th></th>
<th>Req.3</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rank</td>
<td>Weight</td>
<td>R*W</td>
<td>Rank</td>
<td>Weight</td>
<td>R*W</td>
<td>Rank</td>
<td>Weight</td>
<td>R*W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alt. 1</td>
<td>1</td>
<td>10%</td>
<td>0.1</td>
<td>1</td>
<td>10%</td>
<td>0.1</td>
<td>2</td>
<td>10%</td>
<td>0.2</td>
</tr>
<tr>
<td>Alt. 2</td>
<td>5</td>
<td>50%</td>
<td>2.5</td>
<td>4</td>
<td>40%</td>
<td>1.6</td>
<td>4</td>
<td>20%</td>
<td>0.8</td>
</tr>
<tr>
<td>Alt. 3</td>
<td>4</td>
<td>20%</td>
<td>0.8</td>
<td>3</td>
<td>20%</td>
<td>0.6</td>
<td>3</td>
<td>30%</td>
<td>0.9</td>
</tr>
<tr>
<td>Alt. 4</td>
<td>3</td>
<td>20%</td>
<td>0.6</td>
<td>5</td>
<td>30%</td>
<td>1.5</td>
<td>5</td>
<td>40%</td>
<td>2.0</td>
</tr>
<tr>
<td>Σ</td>
<td>13</td>
<td>100%</td>
<td>4.0</td>
<td>13</td>
<td>100%</td>
<td>3.8</td>
<td>14</td>
<td>100%</td>
<td>3.9</td>
</tr>
</tbody>
</table>

As a result, a software used in a PC with Microsoft Windows environment is the best alternative.
CHAPTER IV APPROACH

The approach of this study uses a computer language and interface to implement a numerical solution called Runge-Kutta method. The user would be able to learn the numerical approach by using the program.

IV.1 Structure of the Program

Since the software package was developed under Microsoft Visual Basic 3.0, the program was a combination of nine interface modules called forms. Each of these forms contains several subroutines and functions that were written in Basic language to perform different tasks.

The first form displays the front end of the application and the first level of menu options. Some of these options are disabled when a user starts the program because the program requires input data either from a previously saved data file or from on screen input. The input screen is the second form which contains many input fields of the application. There are many other forms involved in the application. Table IV.1 is a list of forms and their associated subroutines and functions:
Table IV.1 Form Listing

<table>
<thead>
<tr>
<th>Form Name</th>
<th>Description</th>
<th>Major subroutines</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSSP.FR M</td>
<td>Front End</td>
<td>Load_click()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NewF_Click()</td>
</tr>
<tr>
<td>NEWQUEUE.FR M</td>
<td>Data Input</td>
<td>OKinput_Click()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>commSave_Click()</td>
</tr>
<tr>
<td>DDEQUATI.FR M</td>
<td>Show Differential-Differences Equations</td>
<td>ShowDD_Click()</td>
</tr>
<tr>
<td>TRMATRIX.FR M</td>
<td>Show Transition Rate Matrix</td>
<td>showTRM_Click()</td>
</tr>
<tr>
<td>MML.FR M</td>
<td>System Illustration</td>
<td>Picture</td>
</tr>
<tr>
<td>MMS.FR M</td>
<td>System Illustration</td>
<td>Picture</td>
</tr>
<tr>
<td>RESULTAL</td>
<td>Show Results</td>
<td>ShowAll_Click()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mnuPrintResult_Click()</td>
</tr>
<tr>
<td>CHART.FR M</td>
<td>Show Graph Results</td>
<td>mnuShowAll_Click()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mnuEsysQ_Click()</td>
</tr>
<tr>
<td>SHOWSELE.FR M</td>
<td>Show Select Graph</td>
<td>SelectOK_Click()</td>
</tr>
<tr>
<td>CSMF.BAS</td>
<td>Program Declarations</td>
<td>Declarations</td>
</tr>
</tbody>
</table>

After the inputs have been completed, the program automatically enables most of the disabled menu options which allow the user to navigate through the application.

Basically, the program can be logically separated into two major parts. The first part is the user's interface and the second part is the implementation of numerical approximation. In the first part, user friendly is the major concern within its development. Microsoft windows 3.1 user's interface standards were the approaches. In the second part, the fourth order Runge-Kutta method was used to solve a set
of differential-difference equations that simulate the system spontaneously.

IV.2 Programming Flow Chart

The entire program was create under Microsoft Visual Basic 3.0 platform. There are eleven different forms which served as user interface. Figure IV.2.1, Figure IV.2.2, Figure IV.2.3, and Figure IV.2.4 are programming flow chart.

IV.3 Runge-Kutta Method

The Runge-Kutta method is one of the most widely used formulas for the numerical solution of ordinary differential equations. The following are some advantages and disadvantages of using this method:

Advantages:
1. Ease in programming.
2. Good stability characteristics.
3. The step size can be changed as desired without any complications.
4. self-starting capability.

Disadvantages:
1. More computer time is required.
2. Local error estimates are somewhat difficult to obtain.

One of the simplest of the Runge-Kutta type formulas is the second order formulas:

\[ Y_{i+1} = Y_i + \Delta t \cdot f(Y^*_{i+1/2}, t_{i+1/2}) \]

Where

\[ Y^*_{i+1/2} = Y_i + \frac{\Delta t}{2} f(Y_i, t_i) \]

and

\[ t_{i+1/2} = t_i + \frac{\Delta t}{2} \]
Note that the formula is not a multistep formula in the usual sense (no values of \( Y \) earlier than \( Y_i \) are required) and hence the formula is self-starting. However, \( f(Y, t) \) is evaluated more than once. One evaluation is at \((Y_i, t_i)\). Then \( Y^*_{i+1/2} \) is obtained. Next
Figure IV.2.1 Programming Flow Chart
Figure IV.2.2 Programming Flow Chart (Cont.)
Figure IV.2.3 Programming Flow Chart (Cont.)
Figure IV.2.4 Programming Flow Chart (cont.)
The intermediate values $Y_{i+1/2}$, $Y_{i+1/2}^{**}$, and $Y_{i+1}$ must be computed in the order given since they are interdependent. This formula requires four evaluations of $f$, which for complicated functions can be quite time consuming. The other order of Runge-Kutta formula are available but are much less commonly used.

The other expression in which the program was developed with the Runge-Kutta method is as follows:

$$w_0 = \alpha \quad \alpha \text{ is initial value}$$

$$k_1 = hf(t_i, w_i)$$

$$k_2 = hf(t_i + h/2, w_i + 1/2 \ k_1)$$

$$k_3 = hf(t_i + h/2, w_i + 1/2 \ k_2)$$

$$k_4 = hf(t_{i+1}, w_i + k_3)$$

$$w_{i+1} = w_i + 1/6 \ (k_1 + 2k_2 + 2k_3 + k_4)$$
The reason for introducing the terminology $k_1$, $k_2$, $k_3$, $k_4$ into the method is to eliminate the successive nesting in the second variable of $f(t,y)$.

IV.4 The solution of using Runge-Kutta Method in The Simulation Model

As per the discussion in the previous chapter, the mathematical model of the M/M/1 or M/M/s queueing system can be expressed as a set of differential-difference equations as follows:

\[
dP_0(t) \quad \frac{d}{dt} = -\lambda P_0(t) + \mu P_1(t)
\]

\[
dP_1(t) \quad \frac{d}{dt} = \lambda P_0(t) - (\lambda + \mu) P_1(t) + c\mu P_2(t)
\]

\[
dP_2(t) \quad \frac{d}{dt} = \lambda P_1(t) - (\lambda + c\mu) P_2(t) + (c+1)\mu P_3(t)
\]

\[
\vdots
\]

\[
dP_{n-1}(t) \quad \frac{d}{dt} = \lambda P_{n-2}(t) - (\lambda + s\mu) P_{n-1}(t) + s\mu P_n(t)
\]

\[
dP_n(t) \quad \frac{d}{dt} = -\lambda P_{n-1}(t) - s\mu P_n(t)
\]

When M/M/1
\[c = 1\]
\[s = 1\]

When M/M/s
\[c = 2, 3, \ldots \text{Number of server}\]
\[s = \text{Number of server}\]
To approximate the solution of the mth-order system of first-order initial-value problems, the following algorithm was employed.

\[ u'_j = f_j(t, u_1, u_2, u_3, \ldots, u_m) \quad j = 1, 2, 3, \ldots, m. \]

\[ a \leq t \leq b \quad u_j(a) = \alpha_j \quad j = 1, 2, 3, \ldots, m. \]

at \((n+1)\) equally spaced number in the interval \([a,b]::\)

INPUT endpoints \(a,b;\) number of equations (maximum number in the queue system); initial values \(\alpha_1, \ldots, \alpha_m.\)

OUTPUT approximations \(w_j\) to \(u_j(t)\) at the \((N+1)\) values of \(t.\)

STEP 1 Set \(h = (b-a)/N;\) \(t=a.\)

STEP 2 For \(j = 1, 2, 3, \ldots, m\) set \(w_j = \alpha_j.\)

STEP 3 OUTPUT \((t, w_1, w_2, \ldots, w_m).\)

STEP 4 For \(i = 1, 2, 3, \ldots, N\) do steps 5-11.

STEP 5 For \(i = 1, 2, \ldots, m\) set

\[ k_{1,j} = h f_j(t, w_1, w_2, \ldots, w_m). \]

STEP 6 For \(j = 1, 2, \ldots, m\) set

\[ k_{2,j} = h f_j(t + h/2, w_1 + 1/2k_{1,1}, w_2 + 1/2k_{1,2}, \ldots, w_m + 1/2k_{1,m}). \]

STEP 7 For \(j = 1, 2, \ldots, m\) set

\[ k_{3,j} = h f_j(t + h/2, w_1 + 1/2k_{2,1}, w_2 + 1/2k_{2,2}, \ldots, w_m + 1/2k_{2,m}). \]

STEP 8 For \(j = 1, 2, \ldots, m\) set

\[ k_{4,j} = h f_j(t + h, w_1 + k_{3,1}, w_2 + k_{3,2}, \ldots, w_m + k_{3,m}). \]

STEP 9 For \(j = 1, 2, \ldots, m\) set

\[ w_j = w_j + (k_{1,j} + 2k_{2,j} + 2k_{3,j} + k_{4,j})/6. \]

STEP 10 Set \(t = a_i + ih.\)
STEP 11 OUTPUT \((t, w_1, w_2, \ldots, w_m)\).

STEP 12 STOP.
CHAPTER V RESULTS

Some examples were used to run the program and the following is a table list of those examples:

Table V.1 Examples For Testing The Software

<table>
<thead>
<tr>
<th>System Type</th>
<th>Max. Number</th>
<th>No. of Server</th>
<th>Arrival Rate</th>
<th>Service Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>M/M/1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>M/M/2</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>M/M/3</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>M/M/3 (a,b)</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>M/M/4 (c,d)</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>M/M/6 (e,f)</td>
<td>24</td>
<td>6</td>
<td>5.45</td>
<td>2.3</td>
</tr>
<tr>
<td>M/M/10 (g,h)</td>
<td>24</td>
<td>10</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

These example were chosen to test the different situations. Typical results consist of the hard copy of the screen of system information, transition rate matrix, differential-difference equations, numerical integration result, and the charts of the result.

The first three examples contain less than the maximum number in the system so the entire set of transition rate matrix and differential-difference equations can be shown on the hard copy. If a model consists of a large number of the maximum number in system, their transition rate matrix and differential-difference equations can only be seen on the computer screen by using a scroll bar, one of the standard Microsoft windows features, to view them.

Three complete sets of hard copy are shown in Figure V.1, V.2, V.3, V.4, and V.5 but, Figure V.6 is some additional result charts for other examples.
TRANSITION RATE MATRIX

<table>
<thead>
<tr>
<th>FROM</th>
<th>TO</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>-3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>-3</td>
<td>0</td>
<td>2</td>
<td>-3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-2</td>
<td>0</td>
</tr>
</tbody>
</table>

DIFFERENTIAL DIFFERENCE EQUATIONS

<table>
<thead>
<tr>
<th>Coeffs</th>
<th>( p_0(t) )</th>
<th>( p_1(t) )</th>
<th>( p_2(t) )</th>
<th>( p_3(t) )</th>
<th>( p_4(t) )</th>
<th>( p_5(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{dp_0(t)}{dt} )</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \frac{dp_1(t)}{dt} )</td>
<td>1</td>
<td>-3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \frac{dp_2(t)}{dt} )</td>
<td>0</td>
<td>1</td>
<td>-3</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \frac{dp_3(t)}{dt} )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>( \frac{dp_4(t)}{dt} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-3</td>
<td>2</td>
</tr>
<tr>
<td>( \frac{dp_5(t)}{dt} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-2</td>
</tr>
</tbody>
</table>

Figure V.1 Information Hard Copy For Example 1
<table>
<thead>
<tr>
<th>Test</th>
<th>12345</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>p(0)</td>
</tr>
<tr>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td>0.500</td>
<td>0.726</td>
</tr>
<tr>
<td>1.000</td>
<td>0.634</td>
</tr>
<tr>
<td>1.500</td>
<td>0.590</td>
</tr>
<tr>
<td>2.000</td>
<td>0.565</td>
</tr>
<tr>
<td>2.500</td>
<td>0.549</td>
</tr>
<tr>
<td>3.000</td>
<td>0.538</td>
</tr>
<tr>
<td>3.500</td>
<td>0.521</td>
</tr>
<tr>
<td>4.000</td>
<td>0.515</td>
</tr>
<tr>
<td>4.500</td>
<td>0.515</td>
</tr>
<tr>
<td>5.000</td>
<td>0.515</td>
</tr>
<tr>
<td>5.500</td>
<td>0.515</td>
</tr>
<tr>
<td>6.000</td>
<td>0.514</td>
</tr>
<tr>
<td>6.500</td>
<td>0.512</td>
</tr>
<tr>
<td>7.000</td>
<td>0.511</td>
</tr>
<tr>
<td>7.500</td>
<td>0.510</td>
</tr>
<tr>
<td>8.000</td>
<td>0.510</td>
</tr>
<tr>
<td>8.500</td>
<td>0.509</td>
</tr>
<tr>
<td>9.000</td>
<td>0.509</td>
</tr>
<tr>
<td>9.500</td>
<td>0.509</td>
</tr>
<tr>
<td>10.000</td>
<td>0.509</td>
</tr>
</tbody>
</table>

*Table V.2 Results Hard Copy For Example 1*
Figure V.2 Result Charts for Example 1
Max. No. in the System: 4

No. of Servers: 2 in Parallel

Arrival Rate (Poisson): 3 Per Hour

Service Rate (Neg. Exp.): 2 Per Hour

System Types: 2 --- M/H/s

Integration Interval: 0.05 Hours

Print Interval: 10

Total Run Time: 10 Hours

User's Name: Test

ID: 12345

TRANSITION RATE MATRIX

<table>
<thead>
<tr>
<th>FROM</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

DIFFERENTIAL DIFFERENCE EQUATIONS

<table>
<thead>
<tr>
<th>Coef.</th>
<th>p0(t)</th>
<th>p1(t)</th>
<th>p2(t)</th>
<th>p3(t)</th>
<th>p4(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dp0(t)/dt = 2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>dp1(t)/dt = 3</td>
<td>-5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>dp2(t)/dt = 0</td>
<td>3</td>
<td>-7</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>dp3(t)/dt = 0</td>
<td>0</td>
<td>3</td>
<td>-7</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>dp4(t)/dt = 0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>-4</td>
<td></td>
</tr>
</tbody>
</table>

Figure V.3 Information Hard Copy For Example 2
<table>
<thead>
<tr>
<th>Test</th>
<th>12345</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>p(0)</td>
</tr>
<tr>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td>0.500</td>
<td>0.387</td>
</tr>
<tr>
<td>1.000</td>
<td>0.267</td>
</tr>
<tr>
<td>1.500</td>
<td>0.227</td>
</tr>
<tr>
<td>2.000</td>
<td>0.216</td>
</tr>
<tr>
<td>2.500</td>
<td>0.202</td>
</tr>
<tr>
<td>3.000</td>
<td>0.199</td>
</tr>
<tr>
<td>3.500</td>
<td>0.197</td>
</tr>
<tr>
<td>4.000</td>
<td>0.197</td>
</tr>
<tr>
<td>4.500</td>
<td>0.196</td>
</tr>
<tr>
<td>5.000</td>
<td>0.196</td>
</tr>
<tr>
<td>5.500</td>
<td>0.196</td>
</tr>
<tr>
<td>6.000</td>
<td>0.196</td>
</tr>
<tr>
<td>6.500</td>
<td>0.196</td>
</tr>
<tr>
<td>7.000</td>
<td>0.196</td>
</tr>
<tr>
<td>7.500</td>
<td>0.196</td>
</tr>
<tr>
<td>8.000</td>
<td>0.196</td>
</tr>
<tr>
<td>8.500</td>
<td>0.196</td>
</tr>
<tr>
<td>9.000</td>
<td>0.196</td>
</tr>
<tr>
<td>9.500</td>
<td>0.196</td>
</tr>
<tr>
<td>10.000</td>
<td>0.196</td>
</tr>
</tbody>
</table>

Table V.3 Results Hard Copy For Example 2
CSSP

Model M/M/s
Arrival rate 3 per hrs
Service rate 2 per hrs

Figure V.4 Result Charts for Example 2
Figure V.5 Information Hard Copy For Example 3
<table>
<thead>
<tr>
<th>Test</th>
<th>12345</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td><strong>P(0)</strong></td>
</tr>
<tr>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td>0.500</td>
<td>0.729</td>
</tr>
<tr>
<td>1.000</td>
<td>0.649</td>
</tr>
<tr>
<td>1.500</td>
<td>0.622</td>
</tr>
<tr>
<td>2.000</td>
<td>0.612</td>
</tr>
<tr>
<td>2.500</td>
<td>0.603</td>
</tr>
<tr>
<td>3.000</td>
<td>0.607</td>
</tr>
<tr>
<td>3.500</td>
<td>0.606</td>
</tr>
<tr>
<td>4.000</td>
<td>0.606</td>
</tr>
<tr>
<td>4.500</td>
<td>0.606</td>
</tr>
<tr>
<td>5.000</td>
<td>0.606</td>
</tr>
<tr>
<td>5.500</td>
<td>0.606</td>
</tr>
<tr>
<td>6.000</td>
<td>0.606</td>
</tr>
<tr>
<td>6.500</td>
<td>0.606</td>
</tr>
<tr>
<td>7.000</td>
<td>0.606</td>
</tr>
<tr>
<td>7.500</td>
<td>0.606</td>
</tr>
<tr>
<td>8.000</td>
<td>0.606</td>
</tr>
<tr>
<td>8.500</td>
<td>0.606</td>
</tr>
<tr>
<td>9.000</td>
<td>0.606</td>
</tr>
<tr>
<td>9.500</td>
<td>0.606</td>
</tr>
<tr>
<td>10.000</td>
<td>0.606</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Time</strong></th>
<th><strong>P(5)</strong></th>
<th><strong>Sum.P</strong></th>
<th><strong>E(sys)</strong></th>
<th><strong>E(Que)</strong></th>
<th><strong>Lost</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0</td>
</tr>
<tr>
<td>0.500</td>
<td>0.000</td>
<td>1.000</td>
<td>0.316</td>
<td>0.000</td>
<td>0</td>
</tr>
<tr>
<td>1.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.433</td>
<td>0.001</td>
<td>0</td>
</tr>
<tr>
<td>1.500</td>
<td>0.000</td>
<td>1.000</td>
<td>0.476</td>
<td>0.002</td>
<td>0</td>
</tr>
<tr>
<td>2.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.493</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>2.500</td>
<td>0.000</td>
<td>1.000</td>
<td>0.499</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>3.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.501</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>3.500</td>
<td>0.000</td>
<td>1.000</td>
<td>0.502</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>4.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.502</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>4.500</td>
<td>0.000</td>
<td>1.000</td>
<td>0.503</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>5.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.503</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>5.500</td>
<td>0.000</td>
<td>1.000</td>
<td>0.503</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>6.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.503</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>6.500</td>
<td>0.000</td>
<td>1.000</td>
<td>0.503</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>7.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.503</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>7.500</td>
<td>0.000</td>
<td>1.000</td>
<td>0.503</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>8.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.503</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>8.500</td>
<td>0.000</td>
<td>1.000</td>
<td>0.503</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>9.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.503</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>9.500</td>
<td>0.000</td>
<td>1.000</td>
<td>0.503</td>
<td>0.003</td>
<td>0</td>
</tr>
<tr>
<td>10.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.503</td>
<td>0.003</td>
<td>0</td>
</tr>
</tbody>
</table>

Table V.4 Results Hard Copy For Example 3
CSSP

Model M/M/s
Arrival rate 1 per hr
Service rate 2 per hrs

Figure V.6 Result Charts for Example 3
Figure V.7 Result Charts For Examples
CHAPTER VI DISCUSSION AND CONCLUSIONS

The power of personal computers is increasing continuously these days. Predictably more and more professional disciplines or scientific researches will be relying on personal computers because of their portability and affordability compared to the large scale and the expensive mainframe computer. That is the reason the CSSP program was developed.

The CSSP Software took a step towards utilizing the user friendly interface on a PC in order to solve the transient period queueing problem which was difficult for many students who took the operations research class in the past to understand. Now, by using CSSP, a student can repeatedly run numerical integration with different data inputs such as different $\lambda$ and $\mu$ without difficulty. Students should have a good chance at learning how to solve queueing problems after some experience with CSSP.

However, CSSP is limited in solving $M/M/1$ and $M/M/S$ queueing models only. The recommendation for future research is to expand its application to include more queueing models such as $M/E_k/s$, $H_k/M/s$, etc.
References


References (cont.)


References (cont.)


Zwahlen, Helmut T. "An Analysis of the Queueing System M/M/1: (d/FIFO) with Time-Dependent Mean Arrival Rate." M.S. thesis, Ohio University, 1968.
APPENDIX
'Global Declarations
Global MaxNo, Max, dataMax, dataPrt As Integer
Global NumServ, Serv, dataServ As Integer
Global Lambda, L, dataLamda As Single
Global Mu, M, dataMu As Single
Global RunTime, t, dataRT As Single
Global TrnsRtMtx() As Single
Global ST, dataST As Integer 'System Type Value
Global Results() As Single
Global k() As Single
Global p1, p2, p3 As Single
Global p() As Single
Global cp() As Single
Global h, datahh As Single 'increment
Global Esys() As Single
Global EQue() As Single
Global Nnl, Nn2, Nn3, Pr As Integer
Global Lost() As Single
Global flag As Integer
Global sp() As Single
Sub Form_Load ()
    Chart.Cls
    Chart.CurrentX = 5000
    Chart.CurrentY = 100
    Chart.Print "CSSP"
    Chart.CurrentX = 300
    Chart.CurrentY = 20
    Chart.Print "p(t)"
    Chart.CurrentX = 8700
    Chart.CurrentY = 6250
    Chart.Print " hrs."
    Chart.Line (500, 200)-(500, 6000)
    Chart.Line -(9000, 6000)
    j# = 0
    For i% = 6000 To 200 Step -550
        Chart.Line (450, i%)-(500, i%)
        Chart.CurrentY = i% - 100
        Chart.Print j#
        j# = j# + .1
    Next i%
    v% = 0
    y# = 0
    For i% = 500 To 9000 Step (8500 / (T * 2))
        Chart.Line (i%, 6000)-(i%, 6050)
        Chart.CurrentY = 6080
        Chart.CurrentX = i% - 100
        If v% Mod 5 = 0 Then
            Chart.Print y#
        Else
            End If
            y# = y# + .5
            v% = v% + 1
    Next i%
End Sub

Sub mnuClose_Click ()
    Chart.Cls
    Chart.Hide
End Sub

Sub mnuEsysQ_Click ()
    Chart.Cls
    Chart.CurrentX = 5000
    Chart.CurrentY = 100
    Chart.Print "CSSP"
    Chart.CurrentX = 300
    Chart.CurrentY = 20
    Chart.Print "E(n)"
    Chart.CurrentX = 8700
    Chart.CurrentY = 6250
    Chart.Print " hrs."
    Chart.Line (500, 200)-(500, 6000)
For i% = 6000 To 200 Step -1000
    Chart.Line (450, i%)-(500, i%)
    Chart.CurrentX = 60
    Chart.CurrentY = i% - 100
    Chart.Print z#
        z# = z# + 1
Next i%

v% = 0
y# = 0
For i% = 500 To 9000 Step (8500 / (T * 2))
    Chart.Line (i%, 6000)-(i%, 6050)
    Chart.CurrentY = 6080
    Chart.CurrentX = i% - 100
    If v% Mod 5 = 0 Then
        Chart.Print y#
    Else
        End If
        y# = y# + .5
        v% = v% + 1
Next i%

Chart.Line (500, 6000)-(550, 6000 - (Esys(i%) * 1000))
j% = 600
For i% = 2 To (T / h)
    Chart.Line -(j%, 6000 - (Esys(i%) * 1000))
        j% = j% + 50
    If j% > 9000 Then
        GoTo Endraw2
    End If
Next i%
Endraw2:
    Chart.CurrentX = 9000
    Chart.CurrentY = (6000 - (Esys(i%) * 1000)) - 100
    Chart.Print "E(sys)"

Chart.Line (500, 6000)-(550, 6000 - (EQue(i%) * 1000))
j% = 600
For i% = 2 To (T / h)
    Chart.Line -(j%, 6000 - (EQue(i%) * 1000))
        j% = j% + 50
    If j% > 9000 Then
        GoTo Endraw3
    End If
Next i%
Endraw3:
    Chart.CurrentX = 9000
    Chart.CurrentY = (6000 - (EQue(i%) * 1000)) - 100
    Chart.Print "E(Que)"

End Sub
Procedure mnuPChart_Click

    Chart.PrintForm
End Procedure

Procedure mnuShowAll_Click

    Chart.Cls
    Chart.CurrentX = 5000
    Chart.CurrentY = 100
    Chart.Print "CSSP"

    Chart.CurrentX = 6000
    Chart.CurrentY = 300
    If ST = 2 Then
        Chart.Print "Model M/M/s"
    Else
        Chart.Print "Model M/M/1"
    End If

    Chart.CurrentX = 6000
    Chart.CurrentY = 500
    Chart.Print "Arrival rate " & L & " per hrs"
    Chart.CurrentX = 6000
    Chart.CurrentY = 700
    Chart.Print "Service rate " & M & " per hrs"
    Chart.CurrentX = 300
    Chart.CurrentY = 20
    Chart.Print "p(t)"
    Chart.CurrentX = 8700
    Chart.CurrentY = 6250
    Chart.Print " hrs."
    Chart.Line (500, 200)-(500, 6000)
    Chart.Line -(9000, 6000)
    z# = 0

    For i% = 6000 To 200 Step -550
        Chart.Line (450, i%)-(500, i%)
        Chart.CurrentX = 60
        Chart.CurrentY = i% - 100
        Chart.Print z#
        z# = z# + .1
    Next i%

    v% = 0
    y# = 0
    For i% = 500 To 9000 Step (8500 / (T * 2))
        Chart.Line (i%, 6000)-(i%, 6050)
        Chart.CurrentX = 6080
        Chart.CurrentY = i% - 100
        If v% Mod 5 = 0 Then
            Chart.Print y#
        Else
            End If
        y# = y# + .5
        v% = v% + 1
    Next i%
For g% = 1 To Max
    Chart.Line (500, 6000)-(550, 6000 - (cp(1, g%) * 5500))
    j% = 600
    For i% = 2 To Int(T / h)
        Chart.Line -(j%, 6000 - (cp(i%, g%) * 5500))
        j% = j% + 50
        If j% > 9000 Then
            GoTo Endraw
        End If
    Next i%
Endraw:

    Chart.CurrentX = 9000
    Chart.CurrentY = (6000 - (cp(i%, g%) * 5500)) - 100
    Chart.Print "n=" & g%

Next g%

    Chart.Line (500, 200)-(550, 6000 - (cp(1, 0) * 5500))
    j% = 600
    For i% = 2 To Int(T / h)
        Chart.Line -(j%, 6000 - cp(i%, 0) * 5500)
        j% = j% + 50
        If j% > 9000 Then
            GoTo Endraw1
        End If
    Next i%
Endraw1:

    Chart.CurrentX = 9000
    Chart.CurrentY = (6000 - (cp(i%, 0) * 5500)) - 100
    Chart.Print "n=0"

End Sub

Sub mnuShowSelect_Click ()
    ShowSelect.Show
End Sub
Sub DDE_Click ()
  DDEquation.Show
End Sub

Function FN1 (T, p1, p2, L, M)
  FN1 = -L * p1 + M * p2
End Function

Function FN2 (T, p1, p2, p3, L, M)
  FN2 = L * p1 - (L + M) * p2 + M * p3
End Function

Function FN3 (T, p1, p2, L, M)
  FN3 = L * p1 - M * p2
End Function

Sub Ft_Click ()
  Chart.Show
End Sub

Sub mnuFt_Click ()
  Chart.Show
End Sub

Sub mnuGraph_Click ()
  Chart.Show
End Sub

Sub NewQ_Click ()
  NewFile.Show
CSSP.FRM - 2
End Sub

Sub System_Click()
End Sub

Sub TRM_Click()
    TRMatrix.Show
End Sub

Sub Exit_Click()
End Sub

Sub Form_Load()
    mnuSystem.Enabled = False
    mnuTRM.Enabled = False
    mnuDDE.Enabled = False
    mnuOutputs.Enabled = False
    flag = 0
End Sub

Sub Load_Click()
    cdlgBox.DialogTitle = "Open CSP File"
    cdlgBox.Filter = "CSP Files (*.CSP)|*.CSP|All Files (*.*)|*.*"
    cdlgBox.FilterIndex = 1
    cdlgBox.Action = DLG_FILE_OPEN
    cdlgBox.Filename = cdlgBox.Filename
    On Error GoTo DinnaOpen
    Open Filename For Input As #1
    Do While (Not EOF(1))
        Input #1, Max
        Input #1, Serv
        Input #1, L
        Input #1, M
        Input #1, T
        Input #1, STT$
        Input #1, h
        Input #1, Pr
    Loop
    Close
    NewFile.Show
    NewFile!MaxNo.Text = Max
    NewFile!NumServ.Text = Serv
    NewFile!Lambda.Text = L
    NewFile!Mu.Text = M
    NewFile!Runtime.Text = T
    NewFile!SystemType.Text = STT$
    NewFile!hh.Text = h
    NewFile!Prt.Text = Pr
    GoTo good
'oops! bad file name
mbType = MB_ABORTRETRYIGNORE Or MB_DEFBUTTON3 Or MB_ICONEXCLAMATION
buttonPressed = MsgBox("Disk Error", mbType, "Cannot Open File") 'what do you want to do?
Select Case buttonPressed
Case IDABORT
  MsgBox "Data is unchanged"
  'confirm nothing happened
  Exit Sub
Case IDRETRY
  Exit Sub
  'stop the world, I want to get off
  Resume
  'and try to open it again
  Filename = InputBox$("Enter new filename:")
  'plug in a new filename by hand
Case IDIGNORE
  Exit Sub
  'no big deal, just ignore it
End Select
good:
  mnuSystem.Enabled = True
  mnuTRM.Enabled = True
  mnuDDE.Enabled = True
  mnuOutputs.Enabled = True
  mnuOutputs.Enabled = True
  mnuChart.Enabled = False
  View.Enabled = True
  Results.Enabled = True

End Sub
Sub mnuChart_Click ()
  Chart.Show
End Sub
Sub mnuDDE_Click ()
  DDequation.Show
End Sub
Sub mnuOutputs_Click ()
  Result.Show
End Sub
Sub mnuSystem_Click ()
  If ST = 1 Then
    S = 1
    MM1.Show
  ElseIf ST = 2 Then
    MMs.Show
  End If
End Sub
Sub mnuTRM_Click ()
  TRMattrix.Show
End Sub

Sub NewF_Click()
    NewFile.Show
    mnuSystem.Enabled = True
    mnuTRM.Enabled = True
    mnuDDE.Enabled = True
    mnuOutputs.Enabled = True
    mnuOutputs.Enabled = True
    mnuChart.Enabled = False
    View.Enabled = True
    Results.Enabled = True
End Sub
Sub CloseDD_Click()
    DDDequation.Hide
    TRMatrix.Hide
    ShowDD.Enabled = True
    For i% = 0 To Max
        TRMatrix!Grid1.Col = i% + 1
        For j% = 0 To Max
            TRMatrix!Grid1.Row = j% + 1
            TRMatrix!Grid1.Text = " "
        Next j%
    Next i%
    Grid1.Row = 0
    For i% = 0 To Max + 1
        Grid1.Col = i%
        Grid1.Text = " "
    Next i%
    Grid1.Col = 0
    For j% = 0 To Max + 1
        Grid1.Row = j%
        Grid1.Text = " "
    Next j%
End Sub

Sub Form_Load()
    ReDim TrnsRtMtx(0 To Max, 0 To Max)
    DDDequation.Cls
    DDDequation!Grid1.Cols = (Max + 3)
    DDDequation!Grid1.Rows = (Max + 3)
    Grid1.Col = 0
    Grid1.Row = 0
End Sub

Sub mnuPDD_Click()
    DDDequation.PrintForm
End Sub

Sub ShowDD_Click()
    DDDequation.Cls
    DDDequation!Grid1.Cols = (Max + 3)
    DDDequation!Grid1.Rows = (Max + 3)
    Grid1.Col = 0
    Grid1.Row = 0
    Grid1.Text = " "
    Grid1.Col = 0
    For j% = 0 To Max + 1
        Grid1.Row = j%
        Grid1.ColWidth(j%) = 1000
        Grid1.Text = "dp" & (j% - 1) & "(t)/dt ="
    Next j%
    Grid1.Row = 0
DDEQUATI.FRM - 2

For j% = 0 To Max + 1
    Grid1.Col = j%
    Grid1.Text = "p" & (j% - 1) & "(t) +"
Next j%

Grid1.Col = 0
Grid1.Row = 0
Grid1.Text = "Coe."

DDequation!Grid1.Cols = (Max + 3)
DDequation!Grid1.Rows = (Max + 3)
c% = 1
For i% = 0 To Max
    DDequation!Grid1.Col = i% + 1
For j% = 0 To Max
    DDequation!Grid1.Row = j% + 1
    If (i% - j%) = 1 Then
        If j% <= Serv Then
            c% = j%
        Else
            c% = Serv
        End If
    TrnsRtMtx(i%, j%) = c% * M
    ElseIf (j% - i%) = 1 Then
        TrnsRtMtx(i%, j%) = L
    ElseIf i% = j% Then
        If j% <= Serv Then
            c% = j%
        Else
            c% = Serv
        End If
    TrnsRtMtx(i%, j%) = -(L + c% * M)
    Else
        TrnsRtMtx(i%, j%) = 0
    End If
    If j% <= Serv Then
        c% = j%
    Else
        c% = Serv
    End If
    TrnsRtMtx(0, 0) = -L
    TrnsRtMtx(1, 0) = M
    TrnsRtMtx(Max, Max) = -(c% * M)
    DDequation!Grid1.Text = TrnsRtMtx(i%, j%)
Next j%
Next i%

ShowDD.Enabled = False

End Sub
Sub CloseP_Click ()
    MM1.Hide
End Sub

Sub Form_Load ()
    Text1.Text = L
    Text2.Text = Max
    Text3.Text = M
End Sub
MMS.FRM - 1

Sub CloseP_Click ()
    MMS.Hide
End Sub

Sub Form_Load ()
    Text1.Text = L
    Text2.Text = Max
    Text3.Text = M
End Sub
NEWQUEUE.FRM - 1

Dim BadInput As Integer
'MsgBox constants
Const MB_ABORTRETRYIGNORE = 2
Const MB_ICONEXCLAMATION = 48
Const MB_DEFBUTTON3 = 512
Const IDABORT = 3
Const IDRETRY = 4
Const IDIGNORE = 5

'Common Dialog Control
'Action Property
Const DLG_FILE_OPEN = 1
Const DLG_FILE_SAVE = 2
Const DLG_COLOR = 3
Const DLG_FONT = 4
Const DLG_PRINT = 5
Const DLG_HELP = 6

'Individual flags
Const CC_RGBINIT = &H16
Const CF_BOTH = &H36
Const CF_EFFECTS = &H1004
Const PD_ALLPAGES = &H06
Const HELP_KEY = &H101
Const HELP_PARTIALKEY = &H105
Const HELP_QUIT = &H2

Sub CancelInput_Click()
    MaxNo = " "
    NumServ = " "
    Lambda = " "
    Mu = " "
    RunTime = " "
    SystemType = "1 --- M/M/1"
    Max = 0
    Serv = 0
    L = 0
    M = 0
    T = 0
    ST = 1

    NewFile.Hide
End Sub

Sub ClearInput_Click()
    MaxNo = " "
    NumServ = " "
    Lambda = " "
    Mu = " "
    RunTime = " "
    SystemType = "1 --- M/M/1"
    Max = 0
    Serv = 0
    L = 0
    M = 0
    T = 0
    ST = 0
    flag = 0
End Sub
Sub commSave_Click()
    BadInput = (Val(MaxNo.Text) = 0) Or (Val(NumServ.Text) = 0) Or (Val(Lambda.Text) = 0) Or (Val(Mu.Text) = 0) Or (Val(RunTime.Text) = 0) Or (Val(SystemType.Text) = 0)
    BadInput1 = (Val(MaxNo.Text) <> Int(MaxNo.Text)) Or (Val(NumServ.Text) <> Int(NumServ.Text))
    If BadInput Or BadInput1 Then
        If Val(MaxNo.Text) = 0 Or (Val(MaxNo.Text) <> Int(MaxNo.Text)) Then
            InData$ = " Max. No. in the System"
            MaxNo.Text = ""
        ElseIf Val(NumServ.Text) = 0 Or (Val(NumServ.Text) <> Int(NumServ.Text)) Then
            InData$ = " Number of Service"
            NumServ.Text = ""
        ElseIf Val(Lambda.Text) = 0 Then
            InData$ = " Arrival Rate"
            Lambda.Text = ""
        ElseIf Val(Mu.Text) = 0 Then
            InData$ = " Service Rate"
            Mu.Text = ""
        ElseIf Val(SystemType.Text) = 0 Then
            InData$ = " System Type"
            SystemType.Text = ""
        ElseIf Val(RunTime.Text) = 0 Then
            InData$ = " Total run Time"
            RunTime.Text = ""
        Else
            MSG$ = "Input" + InData$ + " values are not valid."
            Title$ = "System Parameters"
            MsgBox MSG$, 48, Title$ GoTo again
        End If
    End If
    If Val(SystemType.Text) = 1 Then
        NumServ.Text = 1
    End If
    BadInput1 = (Val(Prt.Text) = 0) Or (Val(Prt.Text) <> Int(Val(Prt.Text)))
    If BadInput1 Then
        MSG$ = "Invalid print interval"
        Title$ = "System Parameters"
        MsgBox MSG$, 48, Title$ Prt.Text = ""
        GoTo again
    Else
        Pr = Val(Prt.Text)
    End If
    BadInput2 = (Val(hh.Text) > .9) Or (Val(hh.Text) < 0)
    If BadInput2 Then
        MSG$ = "Invalid integration interval"
        Title$ = "System Parameters"
        MsgBox MSG$, 48, Title$ hh.Text = ""
        GoTo again
    Else
        h = Val(hh.Text).
    End If
Max = Val(MaxNo.Text)
Serv = Val(NumServ.Text)
L = Val(Lambda.Text)
M = Val(Mu.Text)
T = Val(RunTime.Text)
STT$ = SystemType.Text

If Not BadInput Or BadInput1 Or BadInput2 Or BadInput3 Then

    cdlgBox.DefaultExt = "CSP"
cdlgBoxDialogTitle = "Save CSP File"
cdlgBox.Filter = "CSP Files (*.CSP)|*.CSP|All Files (*.*)|*.*"
cdlgBox.FilterIndex = 1
cdlgBox.Action = DLG_FILE_SAVE 'Show the File Open

dialog

    Filename = cd1gBox.Filename
    Open Filename For Output As #1
    Print #1, Max
    Print #1, Serv
    Print #1, L
    Print #1, M
    Print #1, T
    Print #1, STT$
    Print #1, h
    Print #1, Pr
    Close #1

End If

again:

End Sub

Sub Form_Load()()
    SystemType.AddItem "1 --- M/M/1"
    SystemType.AddItem "2 --- M/M/s"
    parameter.AddItem "1 -Default"
    parameter.AddItem "2 -Change"
    label11.Visible = False
    NumServ.Visible = False
    Label10.Visible = True
    NumServ.Text = 1
    hh.Text = .05
    Prt.Text = 10
End Sub

Sub mnuPrint_Click()
    NewFile.PrintForm()
End Sub

Sub OKinput_Click()
    ReDim TrnsRtMtx(0 To Val(MaxNo.Text), 0 To Val(MaxNo.Text)) As Single
    BadInput = (Val(MaxNo.Text) = 0) Or (Val(NumServ.Text) = 0) Or (Val(Lambda.Text) = 0) Or (Val(Mu.Text) = 0) Or (Val(RunTime.Text) = 0) Or (Val(SystemType.Text) = 0)
    BadInput3 = (Val(MaxNo.Text) <> Int(MaxNo.Text)) Or (Val(NumServ.Text) <> Int(NumServ.Text))
If BadInput Or BadInput3 Then
    If Val(MaxNo.Text) = 0 Or (Val(MaxNo.Text) <> Int(MaxNo.Text)) Then
        InData$ = " Max. No. in the System"
        MaxNo.Text = " "
    ElseIf Val(NumServ.Text) = 0 Or (Val(NumServ.Text) <> Int(NumServ.Text)) Then
        InData$ = " Number of Service"
        NumServ.Text = " "
    ElseIf Val(Lambda.Text) = 0 Then
        InData$ = " Arrival Rate"
        Lambda.Text = " "
    ElseIf Val(Mu.Text) = 0 Then
        InData$ = " Service Rate"
        Mu.Text = " "
    ElseIf Val(SystemType.Text) = 0 Then
        InData$ = " System Type"
        SystemType.Text = " "
    ElseIf Val(RunTime.Text) = 0 Then
        InData$ = " Total run Time"
        RunTime.Text = " "
    Else
        End If
    End If
    MSG$ = "Input" + InData$ + " values are not valid."
    Title$ = "System Parameters"
    MsgBox MSG$, 48, Title$
    GoTo Redo
Else
    End If
End If

If Val(SystemType.Text) = 1 Then
    NumServ.Text = 1
End If

If BadInput1 = (Val(Prt.Text) = 0) Or (Val(Prt.Text) <> Int(Val(Prt.Text))) Then
    MSG$ = "Invalid print interval"
    Title$ = "System Parameters"
    MsgBox MSG$, 48, Title$
    Prt.Text = " "
    GoTo Redo
Else
    Pr = Val(Prt.Text)
End If

If BadInput2 = (Val(hh.Text) > .9) Or (Val(hh.Text) < 0) Then
    MSG$ = "Invalid integration interval"
    Title$ = "System Parameters"
    MsgBox MSG$, 48, Title$
    hh.Text = " "
    GoTo Redo
Else
    h = Val(hh.Text)
End If

Max = Val(MaxNo.Text)
Serv = Val(NumServ.Text)
L = Val(Lambda.Text)
M = Val(Mu.Text)
T = Val(RunTime.Text)
NEWQUEUE.FRM - 5

    ST = Val(SystemType.Text)
    If Not BadInput Or BadInput1 Or BadInput2 Or BadInput3 Then
        NewFile.Hide
    Else
    End If
    Redo:
    End Sub

Sub Parameter_Click ()
    If Val(parameter.Text) = 2 Then
        hh.Visible = True
        hh.Text = " "
        Prt.Visible = True
        Prt.Text = " "
        Label16.Visible = False
        Label17.Visible = False
    Else
        hh.Visible = False
        hh.Text = .05
        Prt.Visible = False
        Prt.Text = 10
        Label16.Visible = True
        Label17.Visible = True
    End If
End Sub

Sub SystemType_Click ()
    If Val(SystemType.Text) = 2 Then
        NumServ.Visible = True
        NumServ.Text = " "
        Label11.Visible = True
        Label10.Visible = False
    Else
        NumServ.Visible = False
        Label11.Visible = True
        NumServ.Text = 1
        NumServ.Visible = False
        label11.Visible = False
    End If
End Sub
RESULTAL.FRM - 1

Function FN1 (p1, p2, L, M)
    FN1 = -L * p1 + M * p2
End Function

Function FN2 (p1, p2, p3, L, M, s%, c%)
    FN2 = L * p1 - (L + s% * M) * p2 + (c% * M) * p3
End Function

Function FN3 (p1, p2, L, M)
    FN3 = L * p1 - (Serv * M) * p2
End Function

Sub Writescript()
    Grid1.Row = 0
    For i% = 2 To Max + 4
        Grid1.Col = i%
        Grid1.ColWidth(i%) = 700
        Grid1.Text = "P" & "(" & (i% - 2) & ")"
    Next i%
    Grid1.ColWidth(0) = 800
    Grid1.Col = Max + 3
    Grid1.Text = "Sum.P"
    Grid1.Col = Max + 4
    Grid1.Text = "E(sys)"
    Grid1.Col = Max + 5
    Grid1.Text = "E(Q)"
    Grid1.Col = Max + 6
    Grid1.ColWidth(Max + 6) = 800
    Grid1.Text = "cum.Lost"
    Grid1.Row = 0
    Grid1.Col = 0
    Grid1.Text = "Time"
End Sub

Sub CloseResult_Click()
    Result.MousePointer = 11
    Result.Cls
    Result.MousePointer = 0
    Result.Hide
End Sub

Sub Form_Load()
    Result.Cls
    mnuPrintResult.Enabled = False
    Result!Grid1.Cols = (Max + 7)
    Result!Grid1.Rows = Int(T / h) + 3
    Grid1.Col = 0
    Grid1.Row = 0
    Writescript
End Sub

Sub mnuPrintResult_Click()
    Printer.Print
RESULTAL.FRM - 2

y% = 0
z% = 4
If Max Mod 6 = 0 Then
    pag% = (Max / 4)
Else
    pag% = Int(Max / 4) + 1
End If
For x% = 1 To pag%
    If 2% > Max Then
        2% = Max
    End If
    Printer.Print "I1Time" & " I1.
    For j% = y% To 2%
        If j% < 10 Then
            Printer.Print p(" & j% & ") & " I1.
        ElseIf j% < 100 Then
            Printer.Print p(" & j% & ") & " I1.
        Else
            Printer.Print p(" & j% & ") & " I1.
        End If
        Next j%
    Printer.Print "E(sys) & " I1.
    Printer.Print "E(Que) & " I1.
    Printer.Print "Lost I1.
    For i% = 0 To (Int(T / h) + 1)
        If i% Mod Pr = 0 Then
            Printer.Print Format$(xx(i%), "0.##0") & " I1.
            For j% = y% To z%
                Printer.Print Format$(cp(i%, j%), "0.##0") & " I1.
            Next j%
            Printer.Print Format$(sp(i%), "0.##0") & " I1.
            Format$(Esys(i%), "0.##0") & " I1.
            Format$(E(Que)(i%), "0.##0") & " I1.
            Int(Lost(i%)) I1.
            Printer.Print I1.
        End If
        Next i%
    End If
Next x%
Printer.EndDoc

End Sub

Sub ShowAll_Click()
    ReDim p(0 To Max) As Single
    ReDim k(10, 1000)
    ReDim cp(0 To Int(T / h) + 1, 0 To Max) As Single
    ReDim Esys(0 To Int(T / h) + 1) As Single
    ReDim E(Que)(0 To Int(T / h) + 1) As Single
    ReDim Lost(0 To Int(T / h) + 1) As Single
    ReDim xx(0 To Int(T / h) + 1) As Single
    ReDim sp(0 To Int(T / h) + 1) As Single
    For i% = 0 To 4
        For j% = 0 To 1000
            k(i%, j%) = 0
        Next j%
    Next i%
RESULTAL.FRM - 3

Next j%
Next i%

x = 0

Resultl.Grid1.Cols = (Max + 7)
Resultl.Grid1.Rows = (Int(T / h) + 3)

Gridl.Col = 0
Gridl.Row = 0
WriteScript
Gridl.Col = 2
Gridl.Row = 1
Gridl.Text = Format$(1, ".##0")

Gridl.Row = 1
For i% = 3 To Max + 6
Gridl.Col = i%
Gridl.Text = Format$(0, ".##0")
Next i%
Gridl.Col = Max + 6
Gridl.Text = 0
Gridl.Col = Max + 3
Gridl.Text = Format$(1, ".##0")

Gridl.Row = 1
Gridl.Col = 0
Gridl.Text = Format$(0#, ".##0")

For i% = 1 To Max
p(i%) = 0
Next i%
p(0) = 1

a% = 2
c% = 1
s% = 1
Gridl.Row = a%

Resultl.MousePointer = 11

For j% = 1 To Int(T / h) + 1
  k(l, 0) = h * FN1(p(0), p(1), L, M)
For i% = 1 To Max - 1
  If ST > 1 Then
    If i% < Serv Then
      s% = i%
c% = i% + 1
    Else
      s% = Serv
c% = Serv
    End If
  Else
    s% = 1
c% = 1
  End If
  k(l, i%) = h * FN2(p(i% - 1), p(i%), p(i% + 1), L, M, s%, c%)
Next i%
k(l, Max) = h * FN3(p(Max - 1), p(Max), L, M)
k(2, 0) = h * FN1(p(0) + 0.5 * k(1, 0), p(1) + 0.5 * k(1, l), L, M)

For i% = 1 To Max - 1
  If ST > 1 Then
    If i% < Serv Then
      s% = i%
      c% = i% + 1
    Else
      s% = Serv
      c% = Serv
    End If
  Else
    s% = 1
    c% = 1
  End If

k(2, i%) = h * FN2(p(i% - 1) + 0.5 * k(1, i% - 1), p(i%) + 0.5 * k(1, i%), p(i% + 1) + 0.5 * k(1, (i% + 1)), L, M, s%, c%)
Next i%

k(2, Max) = h * FN3(p(Max - 1) + 0.5 * k(1, Max - 1), p(Max) + 0.5 * k(1, Max), L, M)

k(3, 0) = h * FN1(p(0) + 0.5 * k(2, 0), p(1) + 0.5 * k(2, l), L, M)

For i% = 1 To Max - 1
  If ST > 1 Then
    If i% < Serv Then
      s% = i%
      c% = i% + 1
    Else
      s% = Serv
      c% = Serv
    End If
  Else
    s% = 1
    c% = 1
  End If

k(3, i%) = h * FN2(p(i% - 1) + 0.5 * k(2, i% - 1), p(i%) + 0.5 * k(2, i%), p(i% + 1) + 0.5 * k(2, (i% + 1)), L, M, s%, c%)
Next i%

k(3, Max) = h * FN3(p(Max - 1) + 0.5 * k(2, Max), L, M)

k(4, 0) = h * FN1(p(0) + k(3, 0), p(1) + k(3, l), L, M)

For i% = 1 To Max - 1
  If ST > 1 Then
    If i% < Serv Then
      s% = i%
      c% = i% + 1
    Else
      s% = Serv
      c% = Serv
    End If
  Else
    s% = 1
    c% = 1
  End If

k(4, i%) = h * FN2(p(i% - 1) + k(3, i% - 1), p(i%) + k(3, i%), p(i% + 1) + k(3, (i% + 1)), L, M, s%, c%)
Next i%
RESULTAL.FRM  5

   End If
   k(4, i%) = h * FN2(p(i% - 1) + k(3, i% - 1), p(i%) + k(3, i%), p
   (i% + 1) + k(3, (i% + 1)), L, M, s%, c%)  
   Next i%
   k(4, Max) = h * FN3(p(Max - 1) + k(3, (Max - 1)), p(Max) + k(3,
   Max), L, M)

   For i% = 0 To Max
      p(i%) = p(i%) + (1 / 6) * (k(1, i%) + 2 * k(2, i%) + 2 * k(3, i%) +
      k(4, i%))
      cp(j%, i%) = p(i%)
      Next i%

   For i% = 0 To Max
      sum! = sum! + p(i%)  
      sp(j%) = sum!
      Next i%

   For i% = 0 To Max
      sys! = sys! + i% * p(i%)  
      Esys(j%) = sys!
      Next i%

   For i% = Serv To Max
      Que! = Que! + (i% - Serv) * p(i%)
      EQue(j%) = Que!
      Next i%

   Los! = Los! + p(Max) * L
   Lost(j%) = Los!

   If j% Mod Pr = 0 Then
      For i% = 0 To Max
         Grid1.Col = i% + 2
         Grid1.Text = Formats(p(i%), "#.##0")
      Next i%
      Grid1.Col = Max + 3
      Grid1.Text = Formats(sp(j%), "0.##0")
      Grid1.Col = Max + 4
      Grid1.Text = Formats(sys!, "#.##0")
      Grid1.Col = Max + 5
      Grid1.Text = Formats(Que!, "#.##0")
      Grid1.Col = Max + 6
      Grid1.Text = Int(Los!)

      Grid1.Col = 0
      x = 0 + j% * h  
      Grid1.Text = Formats(x, "#.##0")
      xx(j%) = x
      a% = a% + 1
      Grid1.Row = a%
   End If
   Next j%
RESULTAL.FRM - 6

    cp(0, 0) = 1
    sp(0) = 1
    Result.MousePointer = 0
    CSSP!mnuChart.Enabled = True
    mnuPrintResult.Enabled = True
End Sub
Sub n1_Change ()
    BadInput = (Val(n1.Text) > Max) Or (Val(n1.Text) < 0)
    If BadInput Then
        Msg$ = "Input out of range."
        Title$ = "System Parameters"
        MsgBox Msg$, 48, Title$
        n1.Text = ""  
        Nn1 = 0
    End If
End Sub

Sub n2_Change ()
    BadInput = (Val(n2.Text) > Max) Or (Val(n2.Text) < 0)
    If BadInput Then
        Msg$ = "Input out of range."
        Title$ = "System Parameters"
        MsgBox Msg$, 48, Title$
        n2.Text = ""  
        Nn2 = 0
    End If
End Sub

Sub n3_Change ()
    BadInput = (Val(n3.Text) > Max) Or (Val(n3.Text) < 0)
    If BadInput Then
        Msg$ = "Input out of range."
        Title$ = "System Parameters"
        MsgBox Msg$, 48, Title$
        n3.Text = ""  
        Nn3 = 0
    End If
End Sub

Sub SelectClear_Click ()
    n1.Text = ""  
    n2.Text = ""  
    n3.Text = ""
    Nn1 = 0
    Nn2 = 0
    Nn3 = 0
End Sub

Sub SelectOK_Click ()
    Static No1(1 To 3) As Integer
    BadInput = (Val(n1.Text) > Max) Or (Val(n2.Text) > Max) Or (Val(n3.Text) > Max)
    If BadInput Then
        Msg$ = "Input out of range."
        Title$ = "System Parameters"
        MsgBox Msg$, 48, Title$
        Nn1 = Val(n1.Text)
        Nn2 = Val(n2.Text)
        Nn3 = Val(n3.Text)
    ShowSelect.Hide
SHOWSELE.FRM - 2

Chart.Cls
Chart.CurrentX = 5000
Chart.CurrentY = 100
Chart.Print "CSSP"
Chart.CurrentX = 6000
Chart.CurrentY = 300
If ST = 2 Then
   Chart.Print "Model M/M/s"
Else
   Chart.Print "Model M/M/1"
End If
Chart.CurrentX = 6000
Chart.CurrentY = 500
Chart.Print "Arrival rate " & L & " per hrs"
Chart.CurrentX = 6000
Chart.CurrentY = 700
Chart.Print "Service rate " & M & " per hrs"

Chart.CurrentX = 300
Chart.CurrentY = 20
Chart.Print "p(t)"
Chart.CurrentX = 8700
Chart.CurrentY = 6250
Chart.Print " hrs."
Chart.Line (500, 200)-(500, 6000)
j# = 0
For i% = 6000 To 200 Step -550
   Chart.Line (450, i%)-(500, i%)
   Chart.CurrentX = 60
   Chart.CurrentY = i% - 100
   Chart.Print j#
      j# = j# + .1
Next i%
v% = 0
y# = 0
For i% = 500 To 9000 Step (8500 / (T * 2))
   Chart.Line (i%, 6000)-(i%, 6050)
   Chart.CurrentX = 6080
   Chart.CurrentY = i% - 100
   If v% Mod 5 = 0 Then
      Chart.Print y#
   Else
      End If
   y# = y# + .5
   v% = v% + 1
Next i%

No(1) = Nn1
No(2) = Nn2
No(3) = Nn3

For b% = 1 To 3
   q% = No(b%)
   Chart.Line (500, 6000)-(550, 6000 - (cp(1, q%) * 5500))
a% = 600
For $i\% = 2$ To $(T / h)$
    Chart.Line -$(a\%, 6000 - (cp(i\%, g\%) \times 5500))$
    $a\% = a\% + 50$
    If $a\% > 9000$ Then
        GoTo Endraw
    End If
Next $i\%$
Endraw:
Chart.CurrentX = 9000
Chart.CurrentY = $(6000 - (cp(i\%, g\%) \times 5500)) - 100$
Chart.Print "n=" & g\%
Next b\%
Chart.Line $(500, 200)-(550, 6000 - (cp(1, 0) \times 5500))$
$a\% = 600$
For $i\% = 2$ To $(T / h)$
    Chart.Line -$(a\%, 6000 - cp(i\%, 0) \times 5500)$
    $a\% = a\% + 50$
    If $a\% > 9000$ Then
        GoTo Endrawl
    End If
Next $i\%$
Endrawl:
Chart.CurrentX = 9000
Chart.CurrentY = $(6000 - (cp(i\%, 0) \times 5500)) - 100$
Chart.Print "n=0"

End Sub
Sub ShowMtx_Click()
    ReDim TrnsRtMtx(0 To Max, 0 To Max) As Single
    For i% = 0 To Max
        For j% = 0 To Max
            TrnsRtMtx(i%, j%) = 2
            Print TrnsRtMtx(i%, j%)
        Next j%
    Next i%
End Sub

Sub Writescrpt()
    Grid1.Row = 0
    For i% = 0 To Max + 1
        Grid1.Col = i%
        Grid1.Text = (i% - 1)
    Next i%
    Grid1.Col = 0
    For j% = 0 To Max + 1
        Grid1.Row = j%
        Grid1.Text = (j% - 1)
    Next j%
    Grid1.Col = 0
    Grid1.Row = 0
    Grid1.Text = "Stage"
End Sub

Sub CloseMnu_Click()
    TRMatrix.Hide
    ShowTRM.Enabled = True
    For i% = 0 To Max
        For j% = 0 To Max
            TRMatrix!Grid1.Col = i% + 1
            TRMatrix!Grid1.Row = j% + 1
            TRMatrix!Grid1.Text = ""
        Next j%
    Next i%
    Grid1.Row = 0
    For i% = 0 To Max + 1
        Grid1.Col = i%
        Grid1.Text = ""
    Next i%
    Grid1.Col = 0
    For j% = 0 To Max + 1
        Grid1.Row = j%
        Grid1.Text = ""
    Next j%
End Sub

Sub Form_Load()
    ReDim TrnsRtMtx(0 To Max, 0 To Max)
    TRMatrix.Cls
    TRMatrix!Grid1.Cols = (Max + 3)
    TRMatrix!Grid1.Rows = (Max + 3)
End Sub
Sub mnuPmtx_Click()
    TRMatrix.PrintForm
End Sub

Sub mnuShowTRM_Click()
    TRMatrix!Grid1.Cols = (Max + 3)
    TRMatrix!Grid1.Rows = (Max + 3)
    Writescr ipt

    c% = 1
    For i% = 0 To Max
        TRMatrix!Grid1.Col = i% + 1
        For j% = 0 To Max
            TRMatrix!Grid1.Row = j% + 1
            If (i% - j%) = 1 Then
                TrnsRtMtx(i%, j%) = L
            ElseIf (j% - i%) = 1 Then
                If j% <= Serv Then
                    c% = j%
                Else
                    c% = Serv
                End If
                TrnsRtMtx(i%, j%) = c% * M
            ElseIf i% = j% Then
                If j% <= Serv Then
                    c% = j%
                Else
                    c% = Serv
                End If
                TrnsRtMtx(i%, j%) = -(L + c% * M)
            Else
                TrnsRtMtx(i%, j%) = 0
            End If
            If j% <= Serv Then
                c% = j%
            Else
                c% = Serv
            End If
            TrnsRtMtx(0, 0) = -L
            TrnsRtMtx(Max, Max) = -(c% * M)
            TRMatrix!Grid1.Text = TrnsRtMtx(i%, j%)
        Next j%
    Next i%
    ShowTRM.Enabled = False
End Sub