THE UNIVERSITY OF MICHIGAN

SYSTEMS ENGINEERING LABORATORY

Department of Electrical Engineering
College of Engineering

SEL Technical Report No. 34

OPTIMUM DESIGN OF COMPUTER DRIVEN DISPLAY SYSTEMS

by

James D. Foley

March 1969

under contract with:

ROME AIR DEVELOPMENT CENTER
Research and Technology Division
Air Force Systems Command
Contract No. AF 30(602)-3953
Griffiss Air Force Base, New York

This report was also a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in The University of Michigan, 1969.

This work was supported by Air Force Contract AF30(602)-3953.

ACKNOWLEDGMENTS

Completion of the four and one-half years of graduate studies leading up to and including this dissertation was made possible by many people. It all started with the fine undergraduate program in Electrical Engineering at Lehigh University, and the inspiration provided by Dean Karakash. Then came two years of course work here at the University of Michigan, with financial support from NSF and NASA, and more inspiration and encouragement, this time from Dean Scott. Next was a very fruitful period of working with computer graphics under Professor Herzog, who brought to my attention the problems which this dissertation attempts to solve. Now, in the Systems Engineering Laboratory, with the direction and guidance of Professor Irani, and the financial support of Rome Air Development Center, contract AF30(602)-3953, I have endeavored to bring systems analysis and optimization techniques to bear on the problem of designing graphics systems. During this period the members of my doctoral committee have been most helpful with their various constructive comments: for their continuing interest I am most grateful.

This final manuscript has been typed by Miss Linda Oakley. Earlier drafts were typed by Miss Sharon Bauerle, Miss Joyce Doneth, and Mrs. Joanne Aichler.

For her continuing support, encouragement, and understanding, the efforts represented by this dissertation are dedicated to my wife.

James Foley

Ann Arbor, Michigan
March, 1969

TO MARYLOU

TABLE OF CONTENTS

Page

LIST OF TABLES ix

LIST OF FIGURES xi

LIST OF SYMBOLS xiv

ABSTRACT xxi

Chapter

I INTRODUCTION 1

1.1 Why Computer Graphics 2
1.2 Historical Development 5
1.3 Typical Graphics Applications 7
1.4 Definitions 9
1.5 The Hardware System 12
1.6 Tradeoffs 16
1.7 Research Objectives 19

II A MATHEMATICAL MODEL OF DISPLAY SYSTEMS 22

2.1 Display System Model 23
2.2 Hardware Specification 28
2.3 Application Specification 30
2.4 Parameter Calculation 37
2.5 Assumptions 46
2.6 Cost 52
2.7 Analysis 54
  2.7.1 Assignments of Interactions 57
  2.7.2 Monotonicity of T 59

III ANALYSIS METHODS 64

3.1 Simulation 64
3.2 Markov Analysis 64
3.3 Qualitative Comparison of Simulation and Markov Analysis 66
3.4 Quantitative Comparison of Simulation and Markov Analysis 67

TABLE OF CONTENTS (Continued)

Chapter | Page

IV OPTIMIZATION PROCEDURE 78

4.1 Problem Formulation 78
4.2 Optimization Algorithm 81

V EVALUATION OF COMPUTING POWER 94

5.1 Displayable Information 95
5.2 Computing Power—Historical Approaches 102
5.3 Computing Power—Display Instruction Mix 105
5.4 Final Analysis 108

VI APPLICATION 116

6.1 Display System Hardware 117
  6.1.1 Data Link 118
  6.1.2 Remote Computer Core Storage 118
  6.1.3 Remote Computer Bulk Storage 121
  6.1.4 Remote Computer—Display Control 123
6.2 Applications 130
  6.2.1 Text Editing 130
  6.2.2 Two-Dimensional Drawing 133
  6.2.3 Three-Dimensional Drawing 133
  6.2.4 General Network Analysis 138
6.3 Optimization Results 145
6.4 Comparison of Best and Worst Display Systems 175
6.5 Interpretation of Results 178
  6.5.1 Cost-Effectiveness 179
  6.5.2 Multiple Versus Single Console Systems 183
  6.5.3 Guidelines 184
  6.5.4 Hardware Aids for Multiplication and Division 187
6.6 Division of Processing 187
6.7 Summary 190

VII CONCLUSION 192

7.1 Review of the Research 192
7.2 Critical Evaluation 193

REFERENCES 195

TABLE OF CONTENTS (Continued)

Appendix Page

A THE PREPROCESSOR PROGRAM, AND ITS INPUT DATA 201

B THE OPTIMIZATION PROGRAMS, AND THEIR INPUT DATA 212

C DATA FROM DISPLAY APPLICATIONS USING IBM 2250 DISPLAY SYSTEM AND MICHIGAN TERMINAL SYSTEM 233

LIST OF TABLES

Table		Page
3-1	Analysis Results for One User (R = 1)	69
3-2	Analysis Results for Two Users (R = 2)	70
3-3	Analysis Results for Three Users (R = 3)	71
3-4	System Parameters Used for RQA and GPSS Analysis	75
3-5	Convergence of Simulation Results	76
4-1	Optimization Statistics	93
5-1	Display Instruction List	107
6-1	Michigan Intrastate Data Transmission Services	119
6-2	Remote Computer Core Storage	120
6-3	Remote Computer Bulk Storage	122
6-4	Remote Computer-Display Control Configurations	125
6-5	Possible Combinations of Display Controls and Display Consoles	126
6-6	Typical Remote Computer-Display Controls Selected for Use in Optimization	128
6-7	Text Editing Interactions	132
6-8	2-D Drawing Interactions	134
6-9	3-D Drawing Interactions	136
6-10	Network Analysis Interactions	139
6-11	Display Instruction Mix	140
6-12	Display Weights Ω_i and Q_{min}	143

Table		Page
6-13	Application Characteristics	144
6-14	Text Editing, One User	146
6-15	Text Editing, Two Users	148
6-16	Text Editing, Three Users	149
6-17	Two-Dimensional Drawing, One User	153
6-18	Two-Dimensional Drawing, Two Users	155
6-19	Two-Dimensional Drawing, Three Users	157
6-20	Three-Dimensional Drawing, One User	160
6-21	Three-Dimensional Drawing, Two Users	162
6-22	Three-Dimensional Drawing, Three Users	164
6-23	Network Analysis, One User	168
6-24	Network Analysis, Two Users	170
6-25	Network Analysis, Three Users	171
6-26	PM for the Most Cost-Effective Display Systems	189
C-1	Data Gathered for IBM 2250 Display Console Used for Graphics	236
C-2	Data Gathered for IBM 2250 Display Console Used as a Teletype	237
C-3	Data Gathered for Random Teletype Users	238
C-4	Data Gathered for Remote Display Terminal	239
C-5	Comparison of Statistics	240

LIST OF FIGURES

Figure		Page
1-1	Display System	15
2-1	Display System Model	24
2-2	Remote Computer and Queue	41
2-3	Remote Computer and Queue: Transformation I	42
2-4	Remote Computer and Queue: Transformation II	43
2-5	Remote Computer and Queue: Transformation III	44
2-6	An Identity	45
3-1	Service Time Distributions	73
3-2	Branching Distributions	74
4-1	Formation of the Set S	85
4-2	Minimization for One User (R=1)	90
4-3	Final Minimization	91
5-1	Alphanumeric Test Pattern	96
5-2	Weather Map Test Pattern	97
5-3	Graph Test Pattern	98
5-4	Architectural Drawing Test Pattern	99
5-5	Electronic Schematic Test Pattern	100
5-6	Balanced Display Configurations	110
5-7	Unbalanced Display Configuration	111

Figure		Page
5-8	Selection of Remote Computer-Display Controls	113
5-9	Example of Selected Remote Computer-Display Controls	114
6-1	Minimum Response Times, Text Editing	151
6-2	Minimum Response Times, Two-Dimensional Drawing	159
6-3	Minimum Response Times, Three-Dimensional Drawing	166
6-4	Minimum Response Times, Network Analysis	173
6-5	Best and Worst Average Response Times	176
6-6	Cost-Effectiveness	180
A-1	Description of Display Applications	203
A-2	Description of Remote Computer-Display Controls	204
A-3	The PREPROCESSOR Program	209
B-1	Main Program	215
B-2	Input-Output Program	216
B-3	Typical Input Data	219
B-4	Optimization Program	221
B-5	Program to Evaluate T with RQA	224
B-6	Subroutines	229
C-1	Summary Data for Remote Display Terminal	244
C-2	Think Time Distribution	245
C-3	Response Time Distribution	246

Figure Page

C-4 Response Time Distribution 247

C-5 Distribution of Total CPU Time per Response Period 248

C-6 Distribution of Total CPU Time per Response Period 249

C-7 Distribution of CPU Processing Interval Times During Response Periods 250

C-8 Distribution of Output Line Lengths 251

C-9 Distribution of Number of CPU Intervals per Response Period 252

NOMENCLATURE

This list contins the symbols which are used with some frequency in the following report.

Symbol	Meaning	Page
π_i	Probability of service request type i	31
τ_i	Time in microseconds needed by a remote computer-display control to execute a display instruction of type i	106
Θ_i^M	Processing time needed by service request type i if performed by main computer	39
Θ_i^R	Processing time needed by service request type i if performed by remote computer	39
Ω_i	Weight applied to display test pattern of type i	95
$n(V)$	Numerical value of the binary sequence V	82
p_i	Probability of accessing file i	34
t_i	Service time for server i	37
t_1^T	Total average processing time per interaction for server 1	39
t_8^T	Total average processing time per interaction for server 8	39
w_i	Probability of executing display instruction type i	106

Symbol	Meaning	Page
x	Fraction of displayed information which differs between display consoles	102
$\underline{A}$	The vector which minimizes $T(R, \underline{X}, \underline{Z})$	89
ARRIVE	Arrival rate of user service requests	30
B_i^M	Bulk storage accesses needed by service request type i if service performed at main computer	31
B_i^R	Bulk storage accesses needed by service request type i if service performed at remote computer	31
C_i	Cost of remote computer-display control i with no core memory	109
$C(\underline{X})$	Total display system cost per month	53
$C'(\underline{X})$	Display system cost per month, exclusive of main computer computation charges	53
C'_{max}	Upper limit on $C'(\underline{X})$	84
CE	Cost-Effectiveness	179
CPUCST	Monthly cost of using main computer, if it were used whenever display terminal is active	52
F	Fraction of time that display terminal attempts to use the main computer	53

Symbol	Meaning	Page
MAXPAG	Maximum number of pages or files of storage needed by display application	34
MSGLTH	Length of messages sent over data link, in bits	36
N	Number of display controls used at display terminal	109
N(.)	Cardinality of a set	88
N_i	Number of display instructions executed by an interaction of type i	31
N^M	Average number of accesses to bulk storage made by interactions assigned to the main computer	39
N^R	Average number of accesses to bulk storage made by interactions assigned to the remote computer	39
NPPPC	Number of display instructions executed by remote computer for preprocessing or post processing	33
NT	Number of types of service requests	31
PACESS(.)	Cumulative probability of accessing a storage file.	35
PAGCST	Monthly cost of storing one file block at the main computer	53

Symbol	Meaning	Page
PD	Probability that bulk storage is accessed by the main computer following completion of a processing interval	40
PDRC	Probability that a storage file is accessed by the remote computer following completion of a processing interval	46
PEND	Probability that processing is ended at the remote computer when a processing interval ends	40
PM	Probability that any interaction is assigned to the main computer	38
PMAIN	Probability that a storage file is stored at the main computer	36
PRD	Given that a storage file is not in core, the probability that it is stored on the remote computer's bulk storage media	46
PREMOT	Probability that a storage file is stored on the remote computer's bulk storage device	35
Q_i	Percentage of standard display test pattern i displayable by a display control	95
Q_{min}	Minimum percentage of test patterns which must be displayable by a display control, for that control to be considered for an application	112

Symbol	Meaning	Page
Q_{Tp}	Percentage of some special display test pattern displayable by a display control	101
QR	Percentage of test patterns displayable on R display consoles by one display control	102
QRN	Percentage of test patterns displayable on R display consoles by N display controls	109
R	Number of display consoles served by a remote computer	23
S	A set of feasible vectors	84
S_i	The i-th vector in S	90
S'	A set of feasible vectors	88
S_i'	The i-th vector in S'	
S^M	The set of indexes of service request assigned to the main computer	38
S^R	The set of indexes of service requests assigned to the remote computer	38
SYSPAG	Number of pages or files needed by the remote computer's core-resident executive system, plus a display file area and working area for each display console	34
T	Average response time of a display system	54
T_i	Service and queueing time of server i	54

Symbol	Meaning	Page
T^M	Average response time of those service requests assigned to the main computer	54
T_i^M	Response time of service request i if it were assigned to the main computer	57
T^R	Average response time of those service requests assigned to the remote computer	55
T_i^R	Response time of service request i if it were assigned to the remote computer	57
TL	A lower bound on T	56
TMIN	The minimum value of T	89
TRIP	The percentage of all bits sent over the data link which are useful information	29
UMC	Instruction execution rate of the main computer	33
UMD	File access rate of the main computer's bulk storage	28
URD	File access rate of the remote computer's bulk storage	28
X	A binary sequence	81
$\underline{X}$	A four-component vector	29
X1	Data link transmission rate, b. p. s.	29
X2	Blocks of remote computer core storage	29

Symbol	Meaning	Page
X3	Blocks of remote computer bulk storage	29
X4	Instruction execution rate of remote computer-display control	29
X^*	A binary sequence	82
$\underline{X}^*$	A four-component vector	83
$\underline{Z}$	A vector describing a display application	56

ABSTRACT

A rigorous analysis of computer-driven display systems is undertaken. The type of system studied consists of a display terminal, which can include a small computer, core memory, bulk memory, and one or more display controls and display consoles. The display terminal is in turn usually connected via data-link to a large time-shared computing system.

To facilitate the analysis, a mathematical model of a general display system is developed. The model's parameters are derived from characteristics of the display system's hardware and of the application implemented on the system. The model is used to predict the average response time which will be experienced by a user of the display system. Included in the model is an objective method of dividing display processing between the main and terminal computers.

So that certain of the model's parameters can be specified, a method of evaluating the computational and display capabilities of a computer and display is developed. The evaluation criteria are also used to eliminate some computer-display controls from consideration for inclusion in a display system.

The response time can be calculated in one of several ways. If there is only one display console, a closed expression is found. With more than one console, queueing can develop. Thus either simulation or queueing analysis can be used. Comparison of these two techniques shows that even though the conditions needed to use a queueing analysis may not exist, its results in this case are quite satisfactory. Also, queueing analysis is considerably less expensive than simulation.

An optimization procedure is developed to find the display system hardware which, for a given application, minimizes average response time subject only to an upper limit on the amount of money to be spent. The optimization is designed to analyze the display system model as infrequently as possible, to save time.

The optimization is used to find optimum display systems for various costs and for four different display applications: text editing, two-dimensional drawing, three-dimensional drawing, and network analysis. The optimum display systems are in turn used to study the cost-effectiveness of various display systems, to determine if single or multiple display systems are less expensive, to develop general design guidelines, to study the necessity of hardware multiplication and division capabilities at the remote computer, and to demonstrate the necessity of the work reported here.

Chapter I

INTRODUCTION

The subject of the study reported here is the systems design of highly-interactive graphical display terminals for time-shared computer systems. The overall goal is to develop insight into how the choice of subsystems for a display system can affect the system's performance, and to develop methods of finding the combination of subsystems which will be optimum for any well-defined display application. Optimum will be defined so as to minimize a display system's response time subject to a cost constraint.

Viewed in a slightly different way, display system design can be thought of as presenting a problem in resource allocation. The resource is a fixed number of dollars, which is allocated to the purchase of display subsystems in a manner which minimizes the total system's response time.

But why so much interest in dollars and response times? The most important answers are first, that display system hardware costs can easily exceed $100,000; therefore, unwise allocation of so much money falls in the serious mistake category; and second, that improper allocation of the dollars can produce a display system whose response time will be shown to be orders-of-magnitude worse than what can be achieved with the optimum allocation.

This is significant because response time must, of course, be a prime concern in the design of any highly-interactive remote access computer system, and is even more important when considering the graphics terminals which often form part of such systems, because fully capitalizing on the potential interaction rates achievable with a graphics terminal demands good response time.

The various sections of this chapter will attempt to justify the use of graphics terminals, present some pertinent historical information, define terms, and give a qualitative idea of the problem to be tackled.

1.1 Why Computer Graphics?

Dynamic interactive graphics display systems are becoming essential components of large current-day information processing complexes. These displays cannot only replace teletypes currently used in time-sharing computer systems, but can also greatly expand the sphere of computers' reach and usefulness. Thus in the future we can anticipate increasing utilization of display systems to facilitate man-machine interactions.

The advantages of graphic input-output devices manifest themselves in two ways. Display terminals, be they used solely for alphnumeric textual material or for more general graphic presentations, provide an ideal match between the computer and its users. That is, a display terminal is able to accept input from a user at slow keyboard speeds, but can also accept computer output at high speeds, and present it to the user as fast or even faster than he is able to read it. This is in contrast to the prevailing use today of various slow typewriter units, whose slow output speeds in many cases severely limit the realization of otherwise possible man-machine interaction rates. A highly desirable side effect of display terminals is a sharp drop in the proliferation of hard copy output. Many computer terminal uses, such as program preparation and browsing through data files, do not need hard output.

The second justification of display terminals is that for many computer users, particularly engineers, graphics is a natural means of communications, and can bridge the broad chasm between the computer and its multitude of present and future users. This chasm exists because, until recently, computer users have been forced to approach computers in a very stilited and unnatural fashion, dictated more by the computer's input-output limitations than by the user's problem solving requirements. Engineers, and others, are now able to engage the computer in a graphical dialogue, oriented toward using the computer as a design and problem solving aid in a straightforward way.

An excellent example of this chasm bridging exists in the area of computer aided electrical network studies, where analysis and synthesis programs have existed for some time [29] . The problem with such programs is that a potential user must make a large initial investment of time by reading a manual (which may be hard for him to understand) and either preparing punched cards for input or learning to use a teletype terminal. Much more appropriate and suitable systems are now being developed, in which a circuit is actually drawn by the user on a CRT (Cathode Ray Tube) display, component values are specified, dependent and independent sources are defined, and certain currents or voltages are requested to be found [12, 13, 14, 50] . Then, in a few seconds, the desired outputs appear as graphs on the CRT. Given the results, the engineer can now either accept the existing design, or modify it. This is the analysis approach. Synthesis techniques, such as fitting a circuit to given amplitude and phase characteristics [56], are also evolving. But whatever the approach, it is now possible to study complicated electrical networks in a matter of minutes, in a manner more convenient than with either teletype-based remote access systems or card-based batch processing systems.

The basic advantage of display systems over teletypes is not so much the speed increase as the convenience and ease of use for the practicing engineers. Teletypes greatly increase the computer's accessibility, but it is display terminals which brings the computer's power and capabilities to engineers, on engineers' terms, in engineers' language, for engineers' purposes.

Circuit analysis is just one of the many areas in which computer aided design is of maximum benefit when implemented with display terminals. Other applications are discussed in Section 1. 3. We will not bring into our discussion graphics applications which are primarily oriented toward film or paper plotting devices.

Now, having given some indication of the type of computer graphics of interest here, we will turn our attention to a few historic matters.

1. 2 Historical Development

> The first large-scale application of display equipment was the SAGE (Semiautomatic Ground Environment) air-defense system, initiated in the fifties. In this system, operators used light guns and function keys to instruct the computer, and monitored the results on display screens [31].

The development work for this was performed, at least in part, by Mr. Jay Forrester and his associates in M. I. T. 's Whirlwind-I Computer Group, where CRT's were used for man-machine communications as early as 1950 [2]. This early involvement of M. I. T. in computer graphics has proven to be very significant: the school and its associated laboratories continue to pace developments in computer graphics, particularly with respect to new hardware, but also in software. In fact, the second significant graphics contribution from M. I. T., following the pioneering work of the early fifties, was the introduction in early 1963 of a comprehensive software package allowing easy, engineering-oriented man-machine interaction. This software, called Sketchpad[54], was developed by Dr. Ivan Sutherland on Lincoln Lab's TX-2 computer, and marked the beginning of serious efforts to develop sophisticated applications software for graphics work. Manufacturers also began developing the types of display hardware required by these emerging applications.

Coming close on the heels of Sketchpad, the announcement of General Motors Research Laboratories' DAC-I (Design Augmented by Computers) System [23] helped give graphics work respectability and acceptance in industry. Both the academic and industrial worlds have continued to develop and use computer graphics since these early experiments.

A great boon to graphics work has been the emergence of large time-shared computer systems, such as project MAC [8], the Multics System [7], SDC's Q32 System [48], and the Michigan Terminal System [57]. Their impact has been to permit relatively economical on-line graphics work. Before time-sharing systems were available, on-line graphics required a powerful dedicated computer; an expensive proposition indeed! Current graphics systems, along with a large time-sharing computer, use either a small cheap dedicated computer, or none at all.

1. 3 Typical Graphics Applications

The common bond linking the many diverse graphics applications of interest here is that they all embrace a highly interactive graphics-oriented dialogue between a graphics console and its user. This is in marked contrast to uses in which the graphics equipment acts primarily as an output device, or as a sophisticated teletypewriter-like device.

A typical currently implemented application, with widespread industrial usefulness, has been described by Prince [45]. It is a system for creating tapes for numerically controlled machine tools. A designer first draws on his CRT display a standard engineering drawing of a part, and then specifies the path, depth of cut, tool type, feed rate, etc. for the various cuts needed to shape the desired part from a solid piece of metal. The required control tape is then automatically generated. This system eliminates the tedious task of numerical control programming, and has the added advantage of immediate visual confirmation of the cutting tools' paths.

Another interesting application is in the electronics industry, speeding the design of artwork for precision masks used in manufacturing integrated circuits [36, 39, 52]. An engineer can lay out on the CRT the various active and passive circuit components required, specify their characteristics, and indicate the desired component interconnections. The computer then calculates the exact pattern dimensions required at each masking level for the resistors, capacitors, diodes, and transistors, sets up interconnection routings, and produces the required masks as output on a film plotter. The engineer can at any time intervene to modify the computed results, thus allowing the interjection of human judgment at any point in the design process; a highly desirable feature.

Other current uses include textile design [42], biomedical research [46], and mathematical computations and curve plotting [63].

Of significant importance to the implementation of new graphics applications is the recent development of graphics programming systems imbedded in compiler-level languages such as FORTRAN [4, 28, 32, 58]. These systems provide both programming ease and some degree of hardware configuration independence, so that, coupled with increasing hardware availability, we can reasonably anticipate a much more rapid growth of display applications than has been seen in the past. One of the potential stumbling blocks in this progress, however, is the problem of inept display system hardware selection, which can result in systems not suited to their application costing too much and performing poorly.

What we need is a more disciplined approach than has been used in the past for matching display applications to the appropriate hardware. With this closing thought in mind, we will turn our attention to the evolution of present day graphics hardware, by first giving some definitions in Section 1.4, and continuing in Section 1.5 with some specific hardware details.

1.4 Definitions

Before proceeding further, it is necessary that we pause and clearly define certain essential terms so as to avoid later confusion. Particular attention should be given the differences between display terminal, display control, display console, and display.

Display Console

A display console is that piece of equipment which presents graphical information to a human. Current technology utilizes a CRT for this purpose. The console may also include a light pen, printer, function keys, a keyboard, or other similar devices.

Display

A display console presents a display on its cathode ray tube. The display consists of lines (vectors), points, and characters (alphanumerics) forming some meaningful picture, such as a mechanical drawing or a circuit diagram.

Display Control

A display control (sometimes called a display generator) is the interface between a display console and an information processing system. As such, it performs several functions, the most important of which is to accept display commands and produce the appropriate voltages (currents) to drive the CRT's deflection plates (yokes). Sophisticated display controls perform other functions, such as sub-routining, conditional skips, jumps, light pen tracking, and matrix multiplication.

Display Terminal

A display terminal consists of display consoles and whatever computer system hardware has as its prime function the support of those consoles. A display terminal's hardware may include up to one computer, bulk storage devices, a data link interface, and I/O devices, and has a minimum of one display console and one display control.

Display terminals are often connected to a large time-shared computer facility not dedicated primarily to use by the terminal. This facility provides back-up computing and storage support to whatever hardware is directly associated with the terminal.

Remote Display Terminal

A display terminal physically separated from the back-up, or main computer facility is remote from that facility: hence the name.

For our purposes, a terminal will be considered to be remote whenever the data link between the back-up computer and the terminal is more expensive than if the terminal were adjacent to the back-up computer.

Remote Computer

A remote (or peripheral or satellite) computer is a computer which is part of a remote display terminal.

Refresh Rate

Refresh rate is the number of times per second that an image is traced on a volatile (non-storage) display surface. In order that the display maintain a steady intensity, a minimum refresh rate must be maintained. This minimum rate depends upon the decay rate of whatever physical phenomenon produces visible light. For CRT displays, this phenomenon is the secondary emission by phosphor bombarded with high energy electrons. Minimum CRT refresh rates vary from 10 to 60, depending upon the type of phosphor utilized.

Flicker Free

A flicker free display is one which is being regenerated at a rate greater than or equal to the minimum refresh rate, so that to the human eye the intensity of the display remains constant over time.

1.5 The Hardware System

DAC-1 and Sketchpad, the two early interactive graphical computer aided design systems, were both connected to somewhat unorthodox (at the time) computer systems. In the case of DAC-1, an IBM 7094 with two independent 32k core banks was used. One core held the batch processing job; the other, the graphics supervisor and applications programs. When display servicing was needed, control was given to the graphics supervisor which, when finished, reverted control back to the batch supervisor. Thus extensive swapping to a disk or drum is eliminated, and the CPU can be productive much of the time.

Sketchpad is implemented on Lincoln Lab's vintage 1956 experimental TX-2 computer, which has 68k of core memory and 64 index registers. Memory reference instructions can be double indexed, allowing a multiprogramming capability to be implemented with relative ease, in a manner similar to current third generation practices. This of course means, as in the case of DAC-1, that the CPU, when not required by the graphics programs, can be involved in other productive data processing. As mentioned earlier, this is virtually an economic necessity when big computers are involved.

The DAC-1 system helped to introduce the idea of a display buffer memory, because a second function of the 7094's added core bank was to hold the instructions which actually drive the CRT display, thus eliminating any decrease in the instruction execution rate for the batch jobs. With the TX-2, both CPU instructions and display instructions are taken from the same core memory, so that competition for core cycles does arise, thereby slowing the CPU execution rate. In TX-2, ten microseconds are taken to transfer an instruction to the display control, which in turn takes from 20 to 100 microseconds to execute the display instruction. During this execution, the TX-2 has access to its core memory to continue its normal program. In effect, then, the display degrades the computer's performance by 9%to 33%without a buffer memory.

Buffer memories are important for a second reason. Whenever a CRT display console is more than several thousand feet distant from its supporting computer, the cost of the required high-speed data transmission facilities becomes prohibitive. This data link is needed to refresh the CRT display rapidly enough so as to avoid flicker. With a buffer memory at the display console site, a less expensive, lower speed data link becomes feasible.

One of the early (1964) display terminals using as a buffer the core memory of a very small computer was Digital Equipment Corporation's Type 340 Precision Incremental Display. Several configurations exist in which the small computer used is a PDP-4 or PDP-7 [6, 9].

Following in 1965 came DEC's 338 Programmed Buffered Display, which included a PDP-8. This was really the first product line graphics terminal capable both of stand-alone operation for unsophisticated work, and use as a peripheral computer and terminal associated with a large time-shared computer for applications demanding either large amounts of bulk storage or computation. Since then many similar products have been introduced by Information Displays Incorporated, Systems Engineering Laboratory and Adage.

Also available are a number of buffered and unbuffered graphic displays, as well as even more alphanumeric displays. While their ranks continue to swell, our attention will be confined to buffered displays, with or without an associated small peripheral computer.

A general display hardware configuration is shown in Figure 1-1. The bulk storage device is included because in situations where a slow data link is used, the storage can potentially be very desirable and convenient.

For many applications the remote computer is sorely needed. John Ward, from M. I. T. 's Electronic Systems Laboratory, remarks [61] that

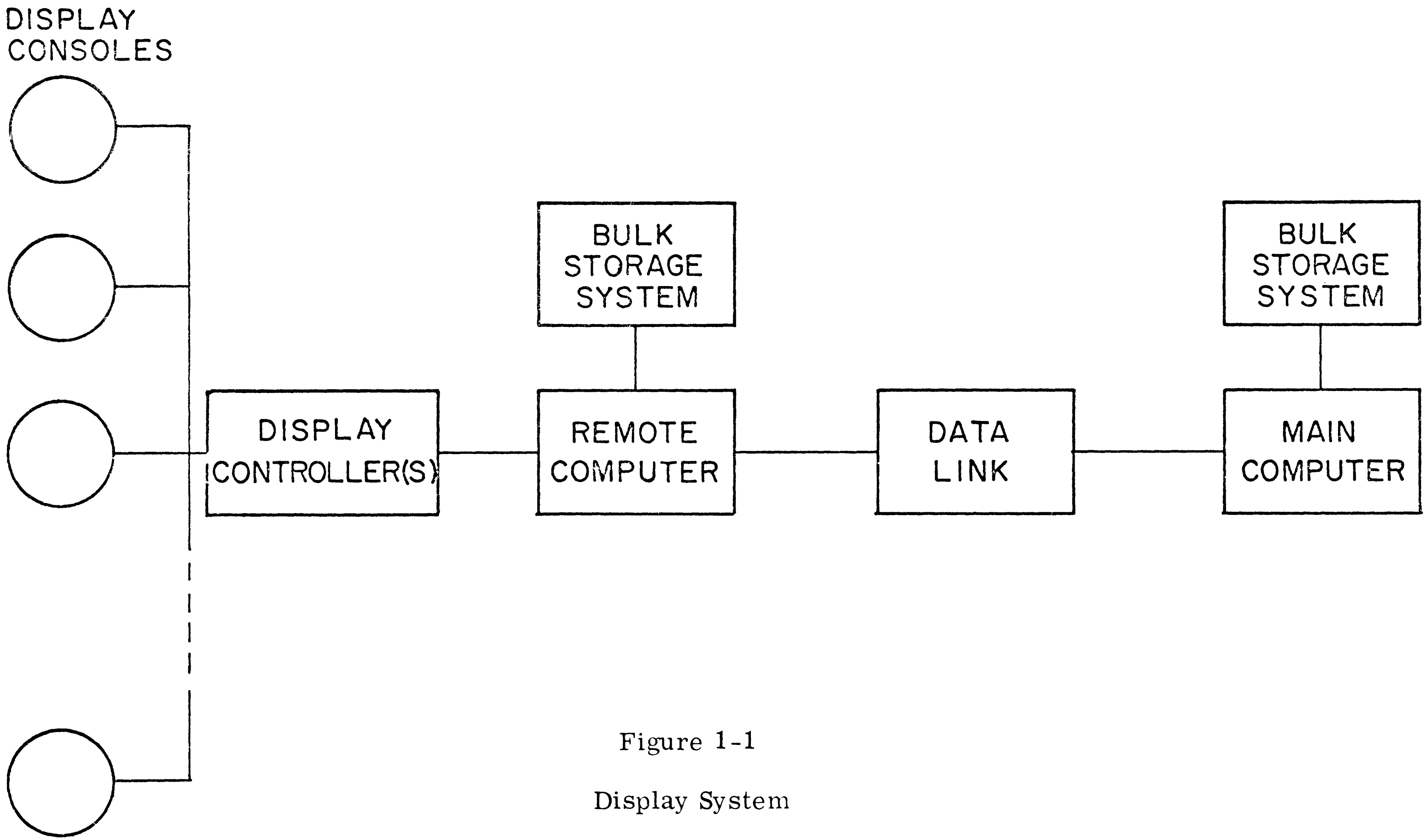


Figure 1-1

Display System

> Our experience over the past two years has indicated that even with the specialized computing capability of the [display] console, the associated real-time general-purpose data processing needed for display operation is an undue burden on the main computer. What is actually needed is a small, inexpensive, satellite computer interposed between the main computer and the display console.

In matching the system in Figure 1-1 to a specific application, many questions arise. These relate to what type of satellite computer should be used, to the data link speed, and to the other hardware components of the display system.

1.6 Trade-offs

There exists a large number of hardware-hardware and hardware-software tradeoffs which can be exploited in specifying a display system for a particular graphics application.

With respect to hardware-hardware tradeoffs, if we wish to maintain a specified response time at the display console and wish to decrease the data link speed, it is necessary to increase the "power" of the remote computer/display control, or the amount of core storage, or the amount of remote bulk storage, in order to compensate for the extra data transmission time. The converse also applies. Increasing remote computer power cuts computation time, while increasing core storage decreases bulk storage accesses, either at the terminal or at the main computer, and increasing remote bulk storage cuts down on data link usage.

In some cases, however, it may not be possible to completely compensate for a lower transmission rate. This will depend both on the decrease in transmission rate and on the relative usage of the four system components. Specifically, a small decrease in transmission rate for an infrequently used data link is far easier to accommodate than a large decrease in a heavily used link.

Similar statements can be made with respect to each of the other system resources: a decrease in any one can be compensated for by increases in one or more of the other resources, within certain limits.

There exist also certain hardware-software tradeoffs, related to the remote computer/display control. These center around the implementation of certain display-oriented functions normally performed at the remote display terminal by either the computer or display control. These functions are discussed below.

When a position indicating device, such as a RAND tablet [11], is used at the display console, it is often necessary to correlate a position with an entity currently being displayed on the CRT. This can be done with software, or with display control hardware which continually compares the current CRT beam position with the indicating device's position [37]. The first method can consume much remote computer time, but costs nothing; the second method takes neither remote computer time nor display control time, but does take money for the extra hardware.

If, on the other hand, a light pen-type entity indicating device is employed, its position will frequently need to be known: this is the familiar light pen tracking problem. Once again, the work can be done with either special purpose hardware built into the display control, or with a program running on the remote computer. A current hardware implementation takes about 10% of the display control's time, decreasing by a like amount the quantity of flicker free material which can be displayed [53]. Software implementations of various pen tracking algorithms do not affect the display, but do require remote computer time to execute.

One of the most demanding display functions is the rotation of a three dimensional object, which requires a matrix multiplication operation of six scalar multiplications and 4 scalar additions for each point and line of the display. Implemented in software, this can be very slow, and can limit the smoothness and rate of dynamic rotation. A first step toward improvement is adding hardware multiplication to the remote computer. A second step is implementation, in the display control, of the actual matrix multiplication. There are two current manifestations of this second possibility. One uses binary rate multipliers, followed by digital addition [53]. The second uses analog multipliers and analog addition [1].

Hardware facilities for display subroutining allow one display list to be used many times in the course of drawing a picture, and therefore avoids needless duplication in core of display instructions. This is all a direct parallel to subroutining for computer programs.

Similar display control hardware-remote computer software tradeoffs exist with respect to problems of dashed lines, blinking lines, transfer of control, recursive subroutining, displaying lines, and displaying alphanumerics.

1.7 Research Objectives

With this multitude of tradeoffs between the various display system components, an important question arises: for a given display system application, and a given dollar cost, what combination of display subsystems will produce the best possible service for display users? The best hardware will produce the fastest, or minimum, average response time experienced by the display system's users.

What is needed for use by display system designers is a rigorous objective method for evaluating the effects upon system cost and response time of the various tradeoffs discussed qualitatively in the previous section. By now the qualitative tradeoffs, which are in fact rather obvious, are well understood. The tradeoffs need to be quantified for the sake of intelligent systems design, because the consequences of using poor systems design are the overloading of some subsystems, underutilization of other subsystems, and decreased productivity for the system's user.

The purpose of the work reported here is to quantify the tradeoffs, and to create a rigorous quantitative approach to display system design. Of prime importance to the work is its continuing emphasis on optimal design. The important contributions of this research are considered to be:

1. Development of a mathematical model for the study of display systems and their applications.
2. Identification of those display system application characteristics which should be known in order to rigorously study the application.
3. Development of an objective approach toward dividing display application computations between the main and remote computers.
4. A comprehensive method to evaluate the capabilities of remote computer-display controls.
5. An optimization scheme to find the best display system of a given cost. This provides a means of studying specific display systems, as well as a way to develop item 7 below.
6. A cost-effectiveness criterion for selecting one of several display systems which are optimum for their respective costs.

7. General guidelines for display system designers.

Chapter II

A MATHEMATICAL MODEL OF DISPLAY SYSTEMS

The central theme of the work reported here is optimum design of display systems. In order that display systems be studied in a rigorous manner, particularly to find optimum display systems, a mathematical model or abstraction of how a display system operates is needed.

To be useful, the model must reflect the varying capabilities of the four hardware components which comprise a general display system, as discussed in Chapter I; data link transmission rate, computation rate of the remote computer and display control, core memory included in the remote computer, and bulk storage associated with the remote computer. The model must also be sensitive to the varying computational, storage, and data transmission requirements of the many different applications which might be implemented with a display system. Furthermore, any explicit or implicit assumptions imbedded in the model must be tenable.

Finally, the model, when appropriately analyzed, must yield some measure of system performance; specifically, the system's response time will be taken to be the most important performance measure.

In the following sections a model which satisfies these requirements will be presented, the process whereby parameters of the model are determined will be discussed, assumptions will be evaluated, and some preliminary analysis concerning response time will be performed.

2.1 Display System Model

Figure 1-1 shows a typical display system with R display consoles being serviced by a single remote computer, which in turn is supported via a data link by a large time-shared computing system. The model, shown in Figure 2-1, represents an abstraction of the flows of information and control which are likely to occur in the total system. Each box in the model represents a server and when $R > 1$, a possible queue of tasks waiting to use the server. As can be seen, servers are the remote computer, its bulk storage, the data link, the main computer, and its bulk storage. It is often the case that several servers in the model refer to the same physical device, performing different functions.

The best way to understand the model's operation is to follow a typical *man-machine interaction* as it passes through the system. An interaction is begun when a user at one of the R display consoles requests service by introducing information into the system. The user does this by typing on an input keyboard, depressing a function button, causing the light pen (or a similar

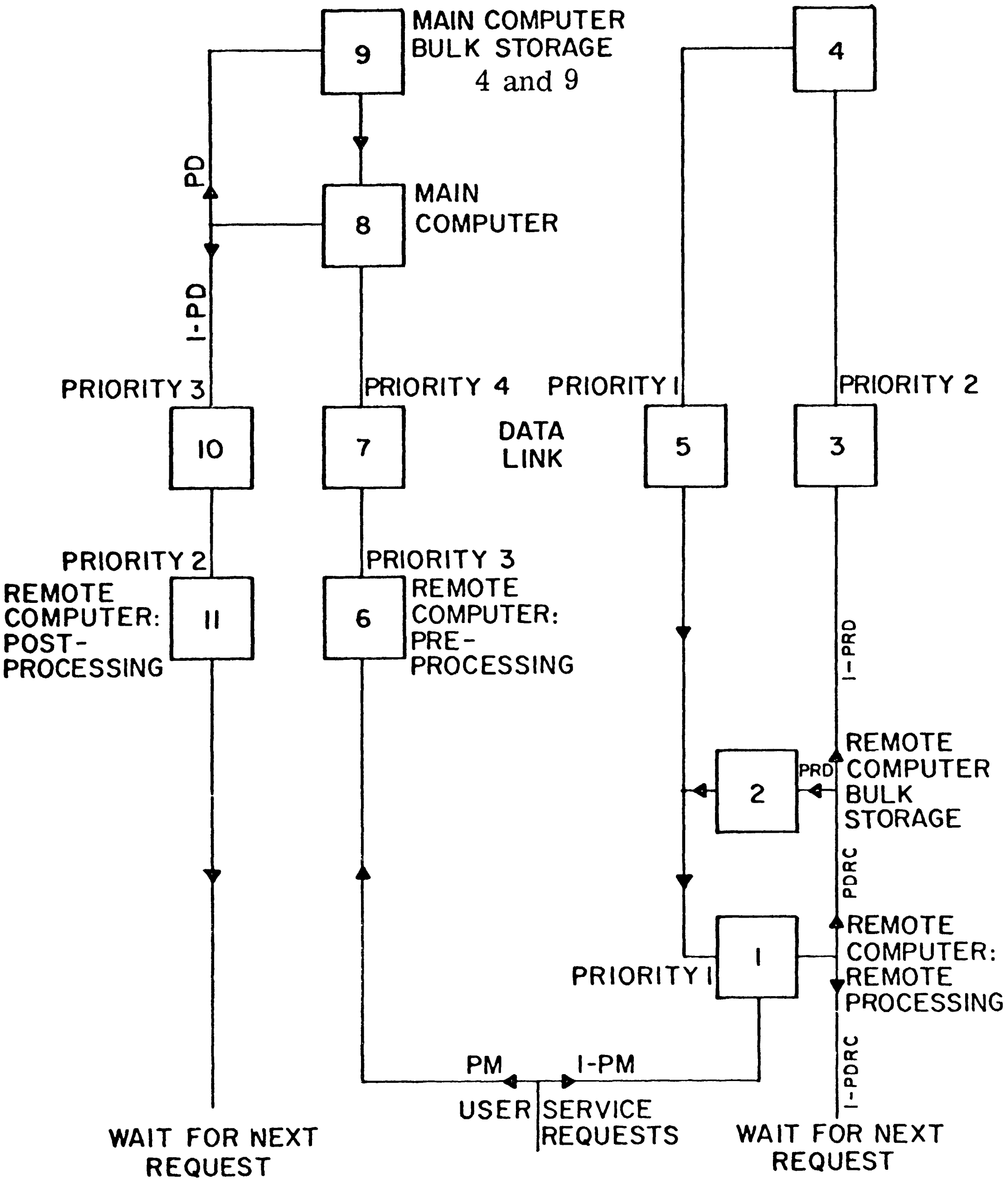


Figure 2-1

Display System Model

device) to sense a point on the display, or some similar action. The resulting service required by the user's action may be so trivial as adding an additional alphanumeric character to the information already in the display, or so difficult as analyzing a complex electrical network.

Having received the input information, the remote computer's interrupt handling software is able to determine, in a time interval usually so short that the process is not modeled, whether the requested service will be performed by the remote computer or the main computer. The main computer will be used if it has been programmed to do what is needed; if not, the remote computer will have been programmed to perform the desired service.

If the main computer is to be used, a task is entered in the queue of tasks waiting to use server 6 (when $R = 1$, queues never exist, so the task always begins service immediately), which is the remote computer in a pre-processing mode of operation. This pre-processing involves preparing a message for transmission to the main computer, giving it all the information needed to perform its task. The work involved may be as simple as requesting transmission of a previously-prepared message, or as complicated as creating a compact description of a display from its display file. Having finished pre-processing, a message is queued for use of the data link, server 7, and is eventually sent to the main computer, after which the job queues for attention by the main computer, server 8.

One of two situations will usually prevail at the main computer. Either display terminal tasks are given priority over other tasks, such as those from teletype terminals, or they are not. In either case, the model's queue 8 contains only display tasks; however, in the priority case, the main computer's service time will be less (possibly substantially less) than when there are no priorities. That is, the fact that non-display tasks may be competing with display tasks for use of the CPU will be treated implicitly in the CPU's service rate, rather than explicitly by having non-display tasks in queue 8. No matter what the priority structure (or lack of it) may be, intervals of main computer service ensue, interspersed with bulk storage accesses to programs and/or data.

Whenever processing ends, a message containing results is queued for transmission to the remote computer via server 10. Following transmission, the task queues to use the remote computer, server 11, for post-processing. During post-processing, results received from the main computer are interpreted and expanded into a display file, after which the user sees his results on the display console, and is able to generate a new service request.

Several general observations concerning the model are appropriate. A very clear distinction is made between the two roles played by the main computer. The first role, that of a processor, is modeled by servers 8 and 9; the second role, that of storage for data and programs for the remote computer, by server 4.

Because several servers are in fact the same physical device, a system of priorities is included in the model. They are indicated in Figure 2-1. For example, if tasks are waiting to use the remote computer in both its pre- and post-processing roles, post-processing is given priority. This is just an extension of the read-before-write philosophy used in scheduling the paging drum of current time-shared computer systems [40, 57]. The goal of course, is to keep response time down. In the case at hand reads are associated with post-processing and writes with pre-processing: since it is the "reads" which bring a task nearer to completion with more certainty than a "write." Indeed, in this example, post-processing completes a task.

Beyond this, however, remote processing (server 1) is given top priority for use of the remote computer. This helps the display terminal users maintain a high interaction rate while doing the ordinary work supported by the remote computer, for which the user expects fast responses. Interactions involving the main computer, for which users are usually prepared to wait a bit longer, are accordingly given lower priority. Priorities for use of the data link are assigned on the same basis, that is, "read before write" with preference given to the remote computer.

Finally, it is important to remember that each of the R users can have only one task in the system at any time. This means that the maximum queue length at any server is R-1 (one will be in service), and that no more than R tasks can simultaneously be in the system.

2.2 Hardware Specification

Determining parameters of the model requires certain information about the hardware system. The parameters defined below are needed.

UMD $\triangleq$ accesses per second which can be made to the Main computer's Disk storage system. For this and the following definition, an access means reading or writing 12,000 bits. If the main computer's filing system is such that storing or retrieving a block of information requires more than one physical access to the disk (such as first accessing a line directory and then accessing the line, as with the Michigan Terminal System [57], then UMD should account for this. Thus, UMD is the rate at which file blocks can be read or written.

URD $\triangleq$ accesses per second which can be made to the Remote computer's Disk storage system. It is assumed that the remote computer's filing system will be very simple, avoiding the necessity of multiple accesses to read or write a file block. Therefore URD is the access rate for a single access.

X1 $\triangleq$ Data link transmission rate, bits per second.

TRIP $\triangleq$ Transfer Rate of Information Percentage, such that TRIB = X1 × TRIP, where TRIB is the actual transfer rate of information bits after accounting for factors such as coding redundancy, error rates, and non-information characters. TRIB is discussed in reference 38.

X2 $\triangleq$ Number of 12, 000 bit blocks of core storage available at the remote computer. A block of storage is defined to be 12, 000 bits as a matter of convenience.

X3 $\triangleq$ Number of 12, 000 bit blocks of bulk storage available at the remote computer.

X4 $\triangleq$ Instruction execution rate of remote computer-display control. The manner in which this can be determined is discussed in Chapter V.

The parameters X1, X2, X3, and X4 each characterize a particular subsystem. The vector $\underline{X}$ = (X1, X2, X3, X4) specifies the essentials of a display system.

The next section will discuss what display application characteristics must be known. Then, knowing the characteristics of both the hardware and the application, the model's parameters will be determinable.

2.3 Application Specification

Characteristics of the particular application being implemented on a display system are needed to completely specify the display system model's parameters. The following characteristics must be known, or estimated. It should be understood that these are average characteristics.

ARRIVE $\triangleq$ Arrival rate of service requests from a user during the period that the system is awaiting commands from the user. This quantity is in fact just the reciprocal of the user's think time. It is clear that as the system's application varies from simple to complex, think time, that is, time required for the user to decide what should be done next, will also vary. Think time also is dependent upon the user's experience with the application and with using the display console.

Each service request which is received from a console user necessitates performing some computations. Depending upon what has been requested, the computations may be very short, or they may be long and involve many bulk storage accesses. For each different type of service which may be desired by the console user, the 4-tuple

(N_i, π_i, B_i^M, B_i^R) must be given. Let there be NT different types of service requests for the application under study.

$N_i \triangleq$ Number of instructions which must be executed to complete a service request of type i. An instruction is not a machine-level instruction, but is rather a higher-level type which will be called "display instructions", and will be precisely defined in Section 5. 3. Any service request can be completed by executing some sequence of display instructions.

$\pi_i \triangleq$ Probability that a service request of type i will be made. The summation of the π_i's is unity.

$B_i^M \triangleq$ Number of accesses (or Branches) to bulk storage which will be required by a type i service request, given that the service request is processed at the Main computer. All of these accesses will be to the main computer's bulk storage.

$B_i^R \triangleq$ Number of accesses (or Branches) to bulk storage which will be made by a type i service request, given that the request is processed at the Remote computer, with only one block of core storage available at the remote computer as a working area into which programs can be brought to execute. These accesses can be to bulk storage at either the main or the remote computer.

B_i^M and B_i^R will in general not be equal. This is primarily because the main computer has more core storage than does the basic remote computer-display control, which as just noted has only one block for a working area. It is therefore quite unlikely that whatever program or data service request type i needs will be in the working area. The main computer in contrast has a large amount of core storage for a working area, so that many programs can be in core (or at least in virtual memory, on a fast drum rather than on slow disk) concurrently. It is thus likely that fewer accesses to bulk storage will have to be made by the main computer than by the remote computer.

It is on the basis of this information that service requests will be assigned to either the main or remote computer. A recent article by Williams [62] contains a general discussion of a graphics system at SDC. It establishes just three types of interactions, which are then assigned to either the main or remote computer depending on the interaction's computational requirements. Section 2. 7. 1 presents a formalized objective method of making these assignments. Once the assignments are made, various parameters of the model can be calculated, as will be done in the following section. First, however, more information is needed.

NPPPC $\triangleq$ Number of instructions performed by the remote computer when in either the pre- or post-processing mode of operation (corresponding to servers 6 and 11 in the model). The meaning of instructions was clarified above.

UMC $\triangleq$ Instruction execution rate for the main computer. Instructions are defined in Section 5. 3. If tasks from the display system are given preemptive priority at the main computer, UMC is the true rate at which display instructions can be performed. If not, UMC must be degraded to account for those periods of time when a display task is eligible to execute but is waiting for a non-display task to complete execution, or when it must wait for core storage to become available. Note that the effect of competition for the CPU by several display tasks is explicitly modeled by use of a queue. It is only competition from tasks not originated by the display system being modeled which must be reflected in UMC.

Another important application characteristic which is manifested in the application programming is the relative and absolute usage of the data and program files associated with the display application. Of specific interest here are all program and data files accessed by the remote computer, regardless of whether the files are stored at the main or remote computer. To delve further into this matter, additional definitions are useful.

MAXPAG $\triangleq$ Maximum number of file Pages, or blocks, needed to store the above-mentioned files, when only one file is stored in a file block. In Section 2. 2 a block of storage was defined as 12, 000 bits.

SYSPAG $\triangleq$ Number of System Pages, or blocks of remote computer core storage used on a permanent resident basis by the executive system, the working area from which programs are executed, and the display file, when one display console is used. Two blocks are added for each additional display console to provide the necessary program area and display file area.

$p_i \triangleq$ Probability of accessing file number i, conditioned on a file access being needed by the remote computer in a display system with only SYSPAG blocks of remote computer core available. Furthermore, let the MAXPAG file blocks be numbered 1, 2, . . . , MAXPAG such that $p_1 \geq \ldots \geq p_{MAXPAG}$. That is, the p_i's form a monotonic nonincreasing sequence $\{p_i\}$.

$PACESS(j) = \sum_{i=1}^{j} p_i \triangleq$ probability of accessing file blocks 1, 2, . . . , j, given that a file access is needed. Then if a display system has X3 blocks of bulk storage at the remote computer and X2 blocks of core storage, it is logical that the X2-SYSPAG file blocks with the highest relative useage, that is, the highest p_i's, be kept in core, the X3 next most frequently used file blocks be kept in remote bulk storage, and the least frequently used blocks be kept at the remote computer. This simply minimizes average file access time by placing frequently used files in a fast storage device and less frequently used files in slower storage devices. With this said, more definitions are possible.

$PCORE = PACESS(X2 - SYSPAG) \triangleq$ Probability that a needed file is in core.

$PREMOT = PACESS\ (X2 + X3 - SYSPAG) - PCORE \triangleq$ Probability that a needed file is in bulk storage at the remote computer.

PMAIN = 1 - PCORE - PREMOT $\triangleq$ Probability that a needed file is in bulk storage at the main computer.

MSGLTH $\triangleq$ Message length, in bits, sent over the data link, corresponding in the model to servers 3, 5, 7, and 10.

Summarizing, the following application parameters are to be estimated.

ARRIVE, the arrival rate of user service requests.

NT, the number of types of service requests.

For i=1, . . . , NT;

N_i, the number of instructions needed to complete service type i.

π_i, the probability of service type i.

B_i^M, the number of bulk storage accesses needed by service type i if performed at the main computer.

B_i^R, the equivalent of B_i^M for service at the remote computer.

NPPPC, the number of instructions needed for pre- or post-processing.

UMC, the main computer's instruction execution rate.

MAXPAG, the maximum number of storage blocks needed by the remote computer.

For i=1, . . . , MAXPAG;

p_i, the probability of accessing file i.

MSGLTH, the length of messages sent over the data link.

At this point it must be very evident that the foregoing parameterized application characterizations may, in fact, be very difficult to estimate. This is true. However, what is equally true is that the better an application and the programming and computational requirements of that application are understood, the easier will the estimating process be, and the better will the parameter estimates be, and the better will the response time estimate be. This model does not give something for nothing. It does not give an estimate of response time for an ill-defined application implemented on an ill-defined hardware display system. On the other hand, the model will hopefully give an accurate estimate of the system response time for a well-defined application.

Having clarified this, attention is turned to the next section, in which the model's parameters are actually found.

2.4 Parameter Calculation

Having described the display system hardware and applications, the model's parameters can now be computed. These parameters consist of the branching probabilities seen in Figure 2-1, the arrival rate of service requests as well as the average service time for each of the 11 servers. These service times will be designated as t_i, i=1, 2, . . . 11.

Certain of the parameters are given explicitly. Thus the arrival rate of service requests, ARRIVE, is known, and $t_2 = 1/URD$, and $t_4 = t_9 = 1/UMD$.

The time taken by the data link to send a message is just the message length, in bits, divided by the effective transmission rate, in bits per second. This rate is just (X1)(TRIP/100), so that

$$t_3 = t_5 = t_7 = t_{10} = \mathrm{MSGLTH}/((\mathrm{X1})(\mathrm{TRIP}/100)). \qquad (2.1)$$

The pre- and post-processing time is the number of instructions to be executed, divided by the remote computer's instruction execution rate, so

$$t_6 = t_{11} = \frac{\mathrm{NPPPC}}{\mathrm{X4}} . \qquad (2.2)$$

The remaining parameters are all dependent upon the division of processing between the main and remote computers. For now it is sufficient to assume some arbitrary division of tasks between the two computers. An optimum division will be discussed in Section 2.7.1.

Now suppose that the set S^R contains the indices of those tasks, or service requests, which are assigned to the remote computer; and S^M, those assigned to the main computer. The properties of S^M and S^R are that $S^M \cap S^R = \phi$, the null set, and $S^M \cup S^R = S$, the set of integers 1, 2, . . . , NT. In words, a task has as its processing unit either the remote or main computer, but not both (exclusive of the pre- and post-processing work), and a task always requires some processing.

Then PM, the probability of using the main computer (the probability that a service request causes a task to use the main computer), is given by

$$PM = \sum' \pi_i, \ i \in S^M.$$

The above properties of S^M and S^R imply that

$$1 - PM = 1 - \sum' \pi_i, \ i \in S^M$$

$$= \sum' \pi_i, \ i \in S^R.$$

From N_i, the processing time per interaction for service request type i is easily seen to be $\Theta_i^M = \frac{N_i}{UMC}$ and $\Theta_i^R = \frac{N_i}{X4}$ for the main or remote computers, respectively. Defining t_1^T and t_8^T to be the total computation time per interaction provided by server 1 (remote computer), given that it is used, and by server 8 (main computer), given that it is used, respectively, it follows that

$$t_1^T = \sum' \frac{\pi_i}{1-PM} \Theta_i^R = \sum' \frac{\pi_i}{1-PM} \frac{N_i}{X4}, \ i \in S^R, \text{ and} \tag{2.3}$$

$$t_8^T = \sum' \frac{\pi_i}{PM} \Theta_i^M, \quad i \in S^M. \tag{2.4}$$

Division by 1- PM in the former and PM in the latter case is needed to form normalized probabilities. In a similar manner, N^R and N^M, the average number of bulk storage accesses per interaction made by the remote computer, given that it is used, and by the main computer, given that it is used, respectively, are

$$N^R = \sum' \frac{\pi_i}{1-PM} B_i^R, \quad i \in S^R, \text{ and}$$

$$N^M = \sum' \frac{\pi_i}{PM} B_i^M, \quad i \in S^M.$$

Having found these intermediate results, the remaining model parameters are determinable. Because there are N^M bulk storage accesses at the main computer, there are $N^M + 1$ processing intervals, the first N^M of which end with bulk storage accesses, and the last of which end not with another bulk storage access, but with the sending of data back to the display terminal. Then $t_8 = \frac{t_8^T}{N^M + 1}$, and the probability of accessing bulk storage is given by $PD = \frac{N^M}{N^M + 1}$. This ratio can be thought of as successful attempts divided by total attempts, where a success is interpreted as a storage access.

The only parameters remaining to be defined are PRDC, PRD, and t_1. These are most easily found by considering Figures 2-2 through 2-6. The rectangle in each figure represents the remote computer, and the expression within each rectangle is the average time used by a task passing through the remote computer. Figure 2-2 shows that the remote computer (server 1 in the model) can complete a job in a time of $\frac{t_1^T}{1 + N^R}$; a direct parallel to the above discussion concerning t_8. The probability that a processing interval completes all needed processing is

$$PEND = \frac{1}{1 + N^R} .$$

If processing is not done, a file access is made either to core with probability PCORE, or to the remote computer's bulk storage with probability PREMOT, or to the main computer's bulk storage with probability PMAIN.

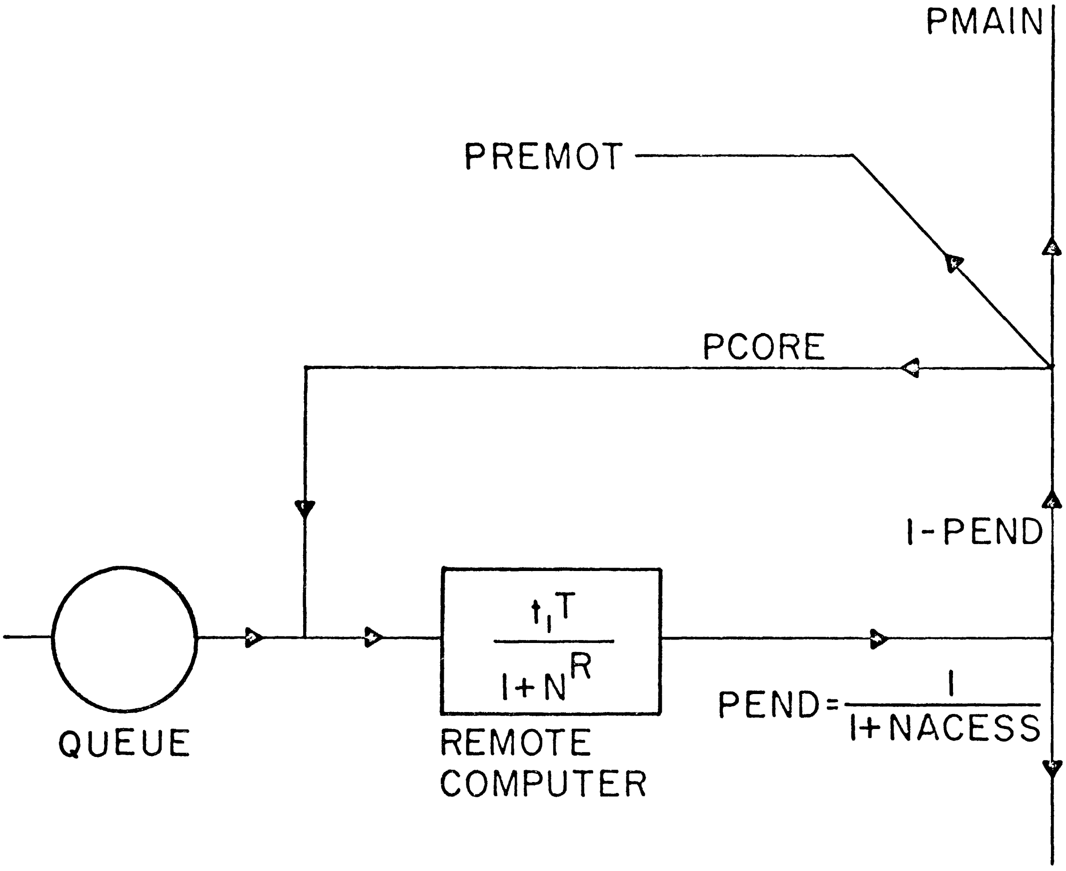


Figure 2-2

Remote Computer and Queue

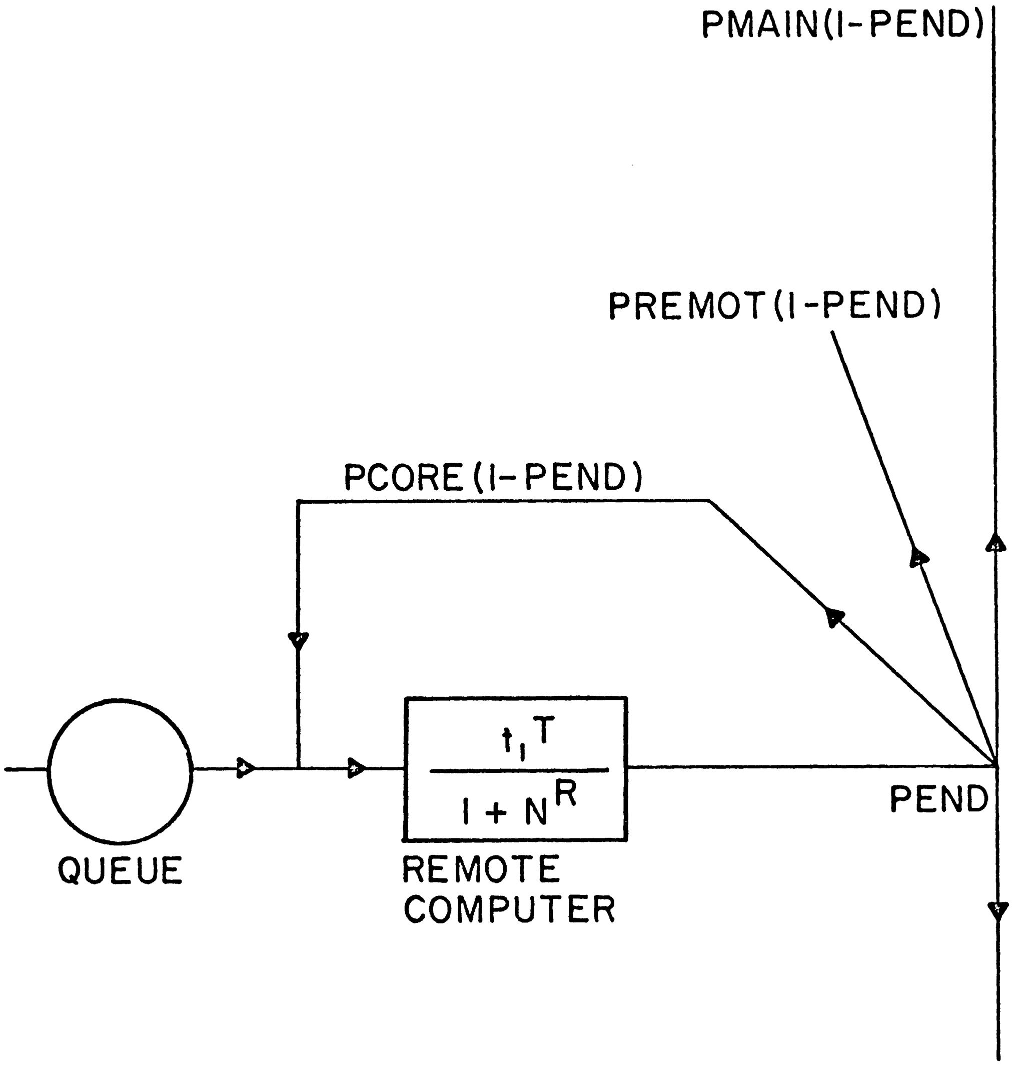


Figure 2-3

Remote Computer and Queue: Transformation I

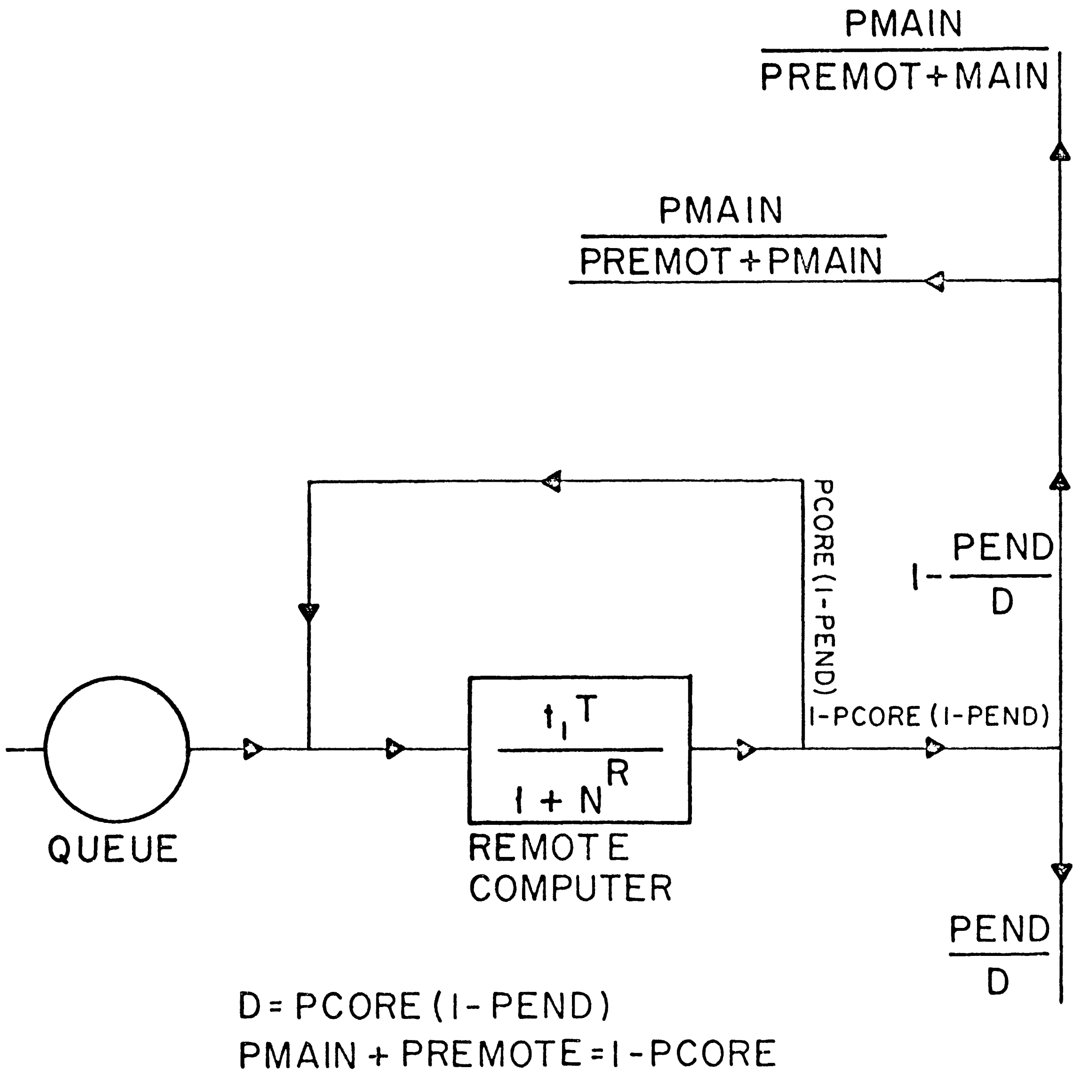
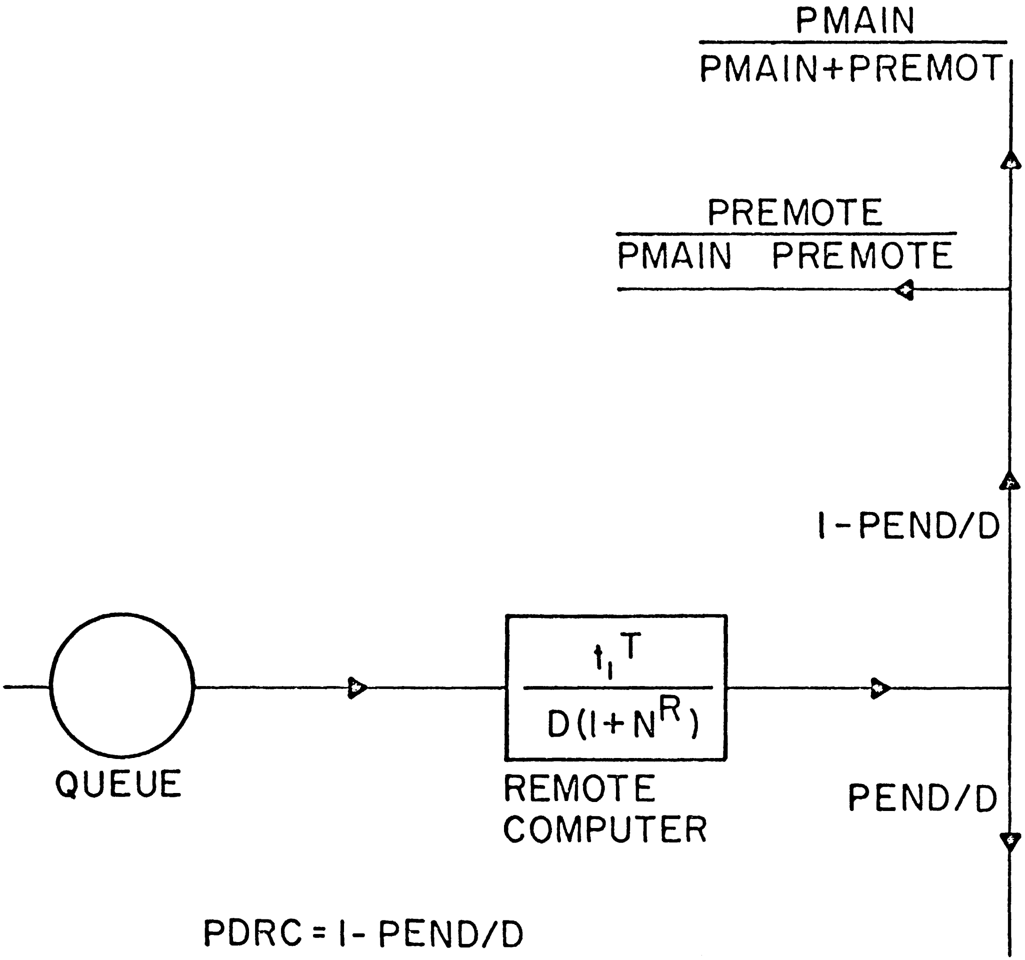


Figure 2-4

Remote Computer and Queue: Transformation II



PDRC = 1- PEND/D
PRD = PREMOTE/(PMAIN + PREMOTE)
D= 1- PCORE (1-PEND)

Figure 2-5

Remote Computer and Queue: Transformation III

Figure 2-6

An Identity

Figure 2-3 represents a merging of the two branching nodes from Figure 2-2 while Figure 2-4 is just a rearrangement of Figure 2-3. Using the identity shown in Figure 2-6 allows the transformation from Figure 2-4 to 2-5, which now is in direct correspondence with the model, so that if D = 1 - PCORE(1 - PEND), then

$$\mathrm{PRDC} = 1 - \mathrm{PEND}/\mathrm{D}, \tag{2. 6}$$

$$\mathrm{PRD} = \mathrm{PREMOT}/(\mathrm{PREMOT} + \mathrm{PMAIN}) \tag{2. 7}$$

$$t_1 = \frac{t_1^T}{D(1 + N^R)}, \tag{2. 8}$$

so that all the parameters are completely specified.

2. 5 Assumptions

Various assumptions have explicitly and implicitly been made during the development of this model. Indeed, further assumptions will be necessary when the model is actually analyzed. At this point, however, only the former variety of assumptions will be discussed. The first of these is that the model requires that all remote computer pre- processing be finished before any information is sent to the main computer. This need not be the case; in a real system, information might be sent as it is extracted from the display file, and the main computer might begin processing received information while the remote computer is still processing as well. Other examples are easily seen. As a result, the response time as predicted by the model will be worse than might realistically be obtained from a system whose programmers make intelligent software design decisions to take advantage of any possible concurrent operations.

This is not as serious as it may seem to be, as a certain amount of concurrency is implicitly imbedded in the model. The most conspicuous example is that of sending or receiving data link messages. The actual transmission would be requested by an application program, but the word-by-word I/O operations would be handled either on an interrupt basis by the executive system, or possibly by a hardware block-transfer facility, while other application programs could be executing.

Another assumption is that all actions associated with a particular service request must be completed before the user can ask for further action. A plausible counterexample is an application in which a user requests that a display be prepared for later offline plotting on a hard copy output device. In most cases there would be no logical reason that the ensuing computations not be performed in a low priority background mode of operation while the display console user proceeds with his graphical work. This assumption can be overcome by not including such cases in the estimates of computational requirements used to find the model's parameters.

The basic structure of the model makes remote disk accesses in the pre- and post- processing operations impossible, while there is certainly nothing inherent in display applications to preclude the desirability of such actions. By eliminating the possibility of such accesses, the model will tend to underestimate the modeled system's response time if the system actually makes the accesses.

The model also allows only one level of bulk storage at the remote computer, specifically head-per-track disks or drums. This limitation is based on two premises. The first is that other reasonably priced bulk storage media which might be used at the remote computer (magnetic tape, for instance) are not as cost-effective as disks or drums, unless very large amounts (more than 800 or 1000 blocks) of on-line bulk storage are needed. The second is that if a large amount of on-line bulk storage is needed, there should still be included a disk or drum for storing the more frequently used files, to help maintain a reasonable response time.

It has been explicitly stated that the estimated parameters forming the application specification are averages. To be more specific and more rigorous, suppose that a display application were actually implemented on a display system, and that the operation of the hardware were closely monitored. Now, if after a very long period of operation, the monitored information were analyzed, and the empirical probability distribution for each of the application parameters were found, the mean of each of the empirical distributions would be used as the corresponding application parameter. In the case of p_i, which is the probability of accessing storage file i, and π_i, which is the probability that a service request is of type i is made, the probability distribution formed is just a count of the accesses and service requests, divided by the total number of accesses and service requests, respectively.

Using information gathered over a very long time span would insure that the mean of the empirical distribution would be very close to the true mean. That is, what is being sought in this way is the long term mean of a statistical process. The process is presumed to be stationary, although in practice this may not be so. For instance, a user's think time may in part depend on what portion of an application he is currently using. In an electrical network application, drawing a simple RLC circuit would probably require less think time per interaction than specifying a forcing function and the output(s) to be plotted. Similarly, differences can exist in the utilization of the remote computer and its memory devices during the different stages of an application. This is why the mean must be found from data taken over a long time span, so that all effects of this sort are taken into full account. This ensures that the means of the various empirical distributions will not be biased towards any particular phase of an application.

Now obtaining data in the above manner is impossible until after a display system is installed and programmed, which is much too late to be of use in determining what type of display system to install and program. Accordingly, a further assumption being made is that the system designer is able to estimate all of the application parameters, using as an aid the above description of how the parameters would be found if it were feasible to do so.

A potentially important phenomenon not fully accounted for by the model is the reduction in "thrashing" which occurs as the amount of remote computer core storage increases. Thrashing occurs to some degree or another when not enough core is available to load a program to be executed, so that it must be loaded piece-by-piece as needed, with the strong possibility that some pieces be loaded many times before execution ends. Thrashing is reduced markedly by increasing core storage. In the model, bulk storage accesses are reduced by increasing core storage because the more active files can then be kept in core. However, there is no explicit mechanism to account for the thrashing reduction which occurs when enough core working area becomes available to hold in its entirety any program which might be initiated by a user interaction. This can be accomplished by making estimates of N_i^R large enough to reflect the thrashing which can occur at the remote computer, and by adjusting PACESS(··) so that the first few file blocks have extremely high useages. Then the addition of the first few blocks of core storage will cause a large decrease in bulk storage accesses, as occurs when thrashing ceases to be a problem.

Another assumption is that the main computer is not needed when the remote computer accesses the main computer's bulk storage. While this is not really true, the processing time needed will be small when compared with the storage access time, and is consequently neglected, thereby reducing the response time estimate.

Why have these assumptions been made? Why can't the model be bigger and more sophisticated so that the assumptions aren't needed? For several reasons: the most important is that as the model becomes larger and more detailed, more and more must be known about the fine details of an application to determine the model's parameters. The model as it exists now is already quite detailed, and is rather demanding in what must be known about the display system application; asking for even more application parameters is most unreasonable and unrealistic.

A second reason for limiting the model's size is limitations of the available analysis tools, i. e. , as the model acquires more detail more computer time is needed for analysis. Because the model will be analyzed many times, the time for a single analysis must be kept within reasonable bounds. That is, a tradeoff exists between the model's complexity and the computer time needed for its analysis. A relatively simple model has been chosen, so that many points in its parameter space can be examined. An excellent example of the converse choice is work done by Nielsen [43], which uses a highly detailed model of the IBM 360/67 and TSS to predict user response time and system loading. Because of the model's detail and resulting long analysis time, only a single very haphazard optimization was possible.

A final point is that the more detailed the model, the fewer will be the display applications which can be represented with the model. Embedding more detail in the model will necessarily begin to bias it toward a particular class of application or computer operating system. This is undesirable. It is believed that the model contains enough detail to reflect the critical differences between applications, yet not so much detail as to scare off potential users.

2.6 Cost

The total cost of the display system includes the hardware's cost, plus the cost of whatever processing is done by the main computer bulk storage used by display system files. Cost is considered in terms of monthly rental charges, because some of the equipment must be leased rather than purchased. Equivalent rentals for purchased hardware are found by amortizing the purchase over 40 months, which is a commonly accepted depreciation period among computer purchasers.

To find main computer usage costs, two quantities are here defined.

CPUCST $\triangleq$ central processing unit cost per month to the display console user if the display system were trying to use the main computer 100% of the time that the display system is in use. CPUCST will be lower for the nonpriority case than for the priority case; in the latter, the display system would actually use the main computer whenever needed, while in the former, because of competition from other jobs, a smaller utilization would be attained. The exact decrease in CPUCST caused by competition from other jobs must be estimated or measured.

PAGCST $\triangleq$ the page, or file block, storage cost per month for files stored at the main computer.

Now if F is the fraction of time the display system is in use that it actually tries to use the main computer.

$$C(\underline{X}) = \text{Total System Cost} = \text{Hardware Cost} + (F)(\text{CPUCST}) + (\text{PAGCST})(\text{MAXPAG} - X2 - X3). \tag{2.9}$$

Also, $C'(\underline{X}) = \text{Hardware cost} + (\text{PAGCST})(\text{MAXPAG} - X2 - X3).$ (2.10)

$C'(\underline{X})$ represents those system costs which can be found without actually analyzing the display system model. That is to say, (F)(CPUCST) has been eliminated because F cannot be quickly determined when there is more than one display console sharing the display system.

Hardware cost is the sum of monthly charges for the data link, remote computer core and bulk storage, and the remote computer-display control (including the display consoles). It is important to note that C' is a monotonic increasing function of $\underline{X}$ as long as the hardware cost of adding either a core or bulk storage page to the remote computer is greater than PAGCST. This will be the case because of economies of scale associated with bulk storage used at the main computer. C' is also nonlinear - linear relations between computer equipment prices and their capabilities seem to be nonexistent.

2.7 Analysis

As mentioned at the beginning of this chapter, the display system model presented here will be used to determine system response time for user service requests. This response time is just the average time delay from when a user's request for service is entered into the system using one of the display console's input devices until the service has been completed, results displayed, and the user permitted to request further service.

If T_i, i=1, 2, . . . , 11 is defined as the average total time in queue for server i, including the actual service time t_i, then it is clear that the response time T is just a weighted sum of the T_i's. That is, the weights will just be the average number of times per interaction that each server is used during the completion of a user's request. That is, if server i were used on the average twice per interaction, its contribution to the total average response time would simply be 2 T_i.

Average response time, as can be determined from Figure 2-2, can be written as

$$T = T^M(PM) + T^R(1 - PM), \tag{2.11}$$

where

$$T^M = T_6 + T_7 + \frac{T_8}{1\text{-}PD} + \frac{T_9(PD)}{1\text{-}PD} + T_{10} + T_{11} \tag{2.12}$$

is the time needed when the main computer is used for processing,

and

$$T^R = \frac{T_1}{1\text{-PDRC}} + \frac{\text{PDRC}}{1\text{-PDRC}} \left[T_2(\text{PRD}) + (T_3 + T_4 + T_5)(1\text{-PRD})\right] \qquad (2.\ 13)$$

is the time needed then the remote computer is used for processing.

Equation 2. 11 simply applies the appropriate weights to the average processing times of the main and remote computers. If the main computer is used for processing, servers 6 7, 10 and 11 are each used once. Therefore T_6, T_7, T_{10}, and T_{11} all have unity multipliers in equation 2. 12. However, the main computer and its bulk storage are used more often. The number of times the main computer is used is given by $1 + \text{PD} + \text{PD}^2 + \text{PD}^3 + \ldots = \frac{1}{1\text{-PD}}$.

Similarly, the number of uses of the main computer's bulk storage is given by $\text{PD} + \text{PD}^2 + \text{PD}^3 + \ldots = \text{PD}(1 + \text{PD} + \text{PD}^2 + \ldots) = \frac{\text{PD}}{1\text{-PD}}$. This accounts for the multipliers of T_8 and T_9.

In equation 2. 13, the multipliers $\frac{1}{1\text{- PDRC}}$ and $\frac{\text{PDRC}}{1\text{-PDRC}}$ are found using the same infinite series, and applied to the remote computer's throughput time and its bulk storage system's throughput time, respectively. The bulk storage throughput time is in turn a weighted sum, with the weights PRD and 1-PRD applied to T_2 and $T_3 + T_4 + T_5$, respectively, reflecting the probability of their usages.

Functionally, response time can be expressed as $T(R, \underline{X}, \underline{Z})$, with R, $\underline{X}$, and $\underline{Z}$ defined below.

$R \triangleq$ number of active display consoles attached to the remote computer. $R \geq 1$.

$\underline{X} = (X1, X2, X3, X4) \triangleq$ description of hardware used in display system, where X1 indicates data link transmission rate; X2, core storage used at the remote computer; X3, bulk storage used at the remote computer; and X4, instruction execution rate for the remote computer-display control.

$\underline{Z} = (z_1, z_2, \ldots, z_n) \triangleq$ description of application implemented by the display system. It will not be necessary to associate each z_i with a particular application parameter, nor to define n.

It is important to note that for any two integers r_1 and r_2 such that $r_1 \leq r_2$, then $T(r_1, \underline{X}, \underline{Z}) \leq T(r_2, \underline{X}, \underline{Z})$. This is really a formalized statement of the obvious, i. e., increasing congestion in the system (increasing the number of system users) also increases response time, all else being equal. A specific and useful instance of this occurs when $r_1 = 1$, $r_2 \geq 1$, for then $T(1, \underline{X}, \underline{Z}) \leq T(r_2, \underline{X}, \underline{Z})$, or $T(1, \underline{X}, \underline{Z}) \leq T(R, \underline{X}, \underline{Z})$. That is, $T(1, \underline{X}, \underline{Z})$ is a lower bound on $T(R, \underline{X}, \underline{Z})$. We thus define

$$T(1, \underline{X}, \underline{Z}) \triangleq TL(R, \underline{X}, \underline{Z}). \qquad (2.14)$$

This result will be used in a later chapter.

Also notice that T is nonlinear in the variables X1, X2, X3, and X4. This will be important in formulating the display system optimization problem in Chapter IV. Two other results will also be needed: they are presented in the following two sections.

2.7.1 Assignment of Interactions

An assignment algorithm which is both simple and reasonable is used to assign interactions to either the main or remote computer for servicing. Define T_i^M as the time that would be required to process a type i interaction if it were assigned to the main computer and T_i^R as the time that would be required to process the same interaction if it were assigned to the remote computer, when just one display console is in use. T_i^M and T_i^R are easily evaluated. Recall that for interaction type i, N_i display instructions must be executed, and either B_i^M main computer or B_i^R remote computer bulk storage accesses must be made. Then, by direct analogy with equations 2.12 and 2.13, and for R = 1,

$$T_i^M = t_6 + t_7 + \frac{t_8}{1-PD} + \frac{t_9(PD)}{1-PD} + t_{10} + t_{11}$$

with $PD = \dfrac{B_i^M}{1+B_i^M}$, and

$$T_i^R = \frac{t_1}{1-PDRC} + \frac{PDRC}{1-PDRC}\left[t_2(PRD) + (t_3 + t_4 + t_5)(1-PRD)\right]$$

with $\quad PDRC = \dfrac{B_i^R}{1+B_i^R}$.

An interaction of type i is assigned to the main computer if $T_i^M < T_i^R$; otherwise, the interaction type is assigned to the remote computer. Clearly this assignment minimizes response time for the single user case, and consequently will be called the optimum assignment. Any other assignment will result in longer response time, and will be suboptimal by the definition being used here.

There are, naturally enough, other definitions of optimal which might be used. Specifically, the current definition does not consider the costs associated with servicing interactions at the main or remote computer. In reality, assigning an interaction type to the main computer will be more expensive than using the remote computer, because the incremental cost of using the remote computer more is zero, while for the main computer it is very definitely positive. This suggests that before assigning an interaction type to the main computer, a relationship such as $T_i^M(1+K) < T_i^R$ or $T_i^M + K < T_i^R$ should be satisfied, where K is some carefully selected positive constant. But now, of course, this introduces an additional parameter into the optimization : a parameter whose proper value is not at all evident. To avoid this additional complexity, then, the main computer will be used whenever $T_i^M < T_i^R$. Thus, in fact, a design decision has been made.

It must be noted that the assignment does not take into account the data base distribution between the main and remote computers. For instance, it would be foolish to store data at the remote computer if the data is used only by programs at the main computer. Therefore the questions of processing assignment and data distribution should be treated together. This has not been done here because of many additional complexities which would arise, and because an unreasonable amount of extra application information would be needed.

When several consoles are active the assignment found as described above is not necessarily optimum, because T_i^M and T_i^R evaluated for one user do not account for the queueing which can develop with multiple users. The problem is that evaluating T_i^M and T_i^R for service request type i with several active users is difficult from a theoretical viewpoint and impossible for practical reasons. In the former case, the assignment of all service requests other than i would have to be known (or assumed) so that the computations needed by request type i would be delayed by the appropriate amount of congestion, whether they were done by the main or remote computer. Presumably, some iterative scheme could be developed using something of a trial-and-error approach to the assignments. In the latter case, as a practical matter, the actual evaluations would be so demanding of computer time as to create an impossible situation. Thus, although the assignment algorithm presented may not, in fact, produce optimum results for the multi-user case, it is the best that can be done.

2.7.2 Monotonicity of T

To demonstrate the desired monotonic nonincreasing property of T with respect to each component of $\underline{X}$, it will be necessary to show that the partial derivatives of T, $\frac{\partial T}{\partial X_i}$, i=1, 2, 3, 4, are less than or equal to zero for positive X_i.

Turning first to X1, the data link transmission rate, it is known from Section 2. 4, equation 2. 1, that $t_3 = t_5 = t_7 = t_{10} = \frac{MSGLTH}{(X1)(TRIP/100)}$. Furthermore T_3 is related to t_3 in such a way that a decrease in t_3 causes a decrease in T_3. Now, $\frac{\partial t_3}{\partial X1} = - \frac{MSGLTH}{(X1)(X1)(TRIP/100}$ so $\frac{\partial T_3}{\partial X1}$ is also negative. The same is true for t_5, t_7, and t_{10}.

Using equations 2. 11, 2. 12, and 2. 13,

$$\frac{\partial T}{\partial X1} = (PM)\left(\frac{\partial T_7}{\partial X1} + \frac{\partial T_{10}}{\partial X1}\right) + (1-PM)\left(\frac{PRDC}{1-PDRC}\right)(1-PRD)\left(\frac{\partial T_3}{\partial X1} + \frac{\partial T_5}{\partial X1}\right) \tag{2. 15}$$

Now each of the four derivatives in equation 2. 15 is negative, so $\frac{\partial T}{\partial X1}$ is negative.

Continuing with X2, the size of the remote computer's core storage, the derivative of T is again found using equations 2. 11, 2. 12, and 2. 13. It is

$$\frac{\partial T}{\partial X2} = (1-PM)N^R(T_2 - T_3 - T_4 - T_5) \frac{\partial}{\partial X2} PACESS \, (X2 + X3 - SYSPAG) \tag{2. 16}$$

The derivative of PACESS (•) is positive, because increasing X2 increases PACESS (•). Thus, in order that 2. 16 be nonpositive, the inequality $T_2 \leq T_3 + T_4 + T_5$ must be satisfied. This is nothing more than requiring that the remote computer's bulk storage, including queueing delays, be faster than the sum of data link transmission times and access to the main computer's bulk storage, including queueing delays. If this were not the case, there would be no justification for remote bulk storage, because using the main computer's bulk storage would be faster. There is, in fact, no difficulty satisfying the inequality, because the movable head disk pack storage units used at the main computer are considerably slower than the head-per-track disks which would be used at the remote computer. Thus, equation 2. 16 is nonpositive, and T is monotonic nondecreasing with with respect to X2.

Turning now to X3, which is the size of the remote computer's bulk storage, equations 2. 11, 2. 12, and 2. 13 yield

$$\frac{\partial T}{\partial X3} = (1\text{-}PM)\frac{PDRC}{1\text{-}PDRC}(T_2 - T_3 - T_4 - T_5)\frac{\partial PRD}{\partial X3} \quad . \qquad (2.\ 17)$$

Now $\frac{\partial PRD}{\partial X3}$ is positive, because increasing the amount of bulk storage increases the probability of its use, which is what PRD happens to be. But, as required in the preceding paragraph, $T_2 \leq T_3 + T_4 + T_5$, so that $T_2 - T_3 - T_4 - T_5$ is nonpositive. Therefore, 2. 17 is also nonpositive, so T is monotonic nonincreasing with respect to X3.

Finally, turning attention to X4, first recall the two equations

$$t_6 = t_{11} = \frac{NPPPC}{X4} \quad , \text{ and} \qquad (2.\ 2)$$

$$t_1 = \frac{t_1^T}{D(1 + N^R)} \quad . \qquad (2.\ 8)$$

Using equation 2. 3 with 2. 8 yields

$$t_1 = \frac{1}{D(1+N^R)} \sum \frac{\pi_i N_i}{(1-PM)(X4)} \qquad i \in S^R \qquad (2.\ 18)$$

Differentiating,

$$\frac{\partial t_1}{\partial X1} = \frac{-1}{D(1+N^R)} \sum \frac{\pi_i N_i}{(1-PM)(X4)^2} \qquad i \in S^R \qquad (2.\ 19)$$

Just as T_3 decreased when t_3 decreased, so too with T_1 and t_1. Therefore, since $\frac{\partial t_1}{\partial X4}$ is negative, so is $\frac{\partial T_1}{\partial X4}$. Also, differentiating 2. 4. 2

$$\frac{\partial t_6}{\partial X4} = \frac{\partial t_{11}}{\partial X4} = -\frac{NPPPC}{(X4)^2} \ . \qquad (2.\ 20)$$

As was the case with T_1 and t_1, so with T_6 and t_6, and T_{11} and t_{11}. Thus, $\frac{\partial T_6}{\partial X4}$ and $\frac{\partial T_{11}}{\partial X4}$ are negative.

Now, again using equations 2. 11, 2. 12, and 2. 13,

$$\frac{\partial T}{\partial X4} = PM\left(\frac{\partial T_6}{\partial X4} + \frac{\partial T_{11}}{\partial X4}\right) + \frac{1-PM}{1-PDRC}\ \frac{\partial T_1}{\partial X4} \qquad (2.\ 21)$$

Each of the derivatives on the right of equation 2. 21 is negative; therefore $\frac{\partial T}{\partial X4}$ is negative and T is monotonic nonincreasing with respect to X4.

The conclusion to be drawn from all this is that, since equations 2. 15, 2. 16, 2. 17, and 2. 21 are all nonpositive, T is monotonic nonincreasing with respect to each component of $\underline{X}$ = (X1, X2, X3, X4).

This means that whenever additional hardware is added to a display system, the system's response time will not increase.

Chapter III

ANALYSIS METHODS

In the preceding chapter a mathematical model of a display system was presented. The model attempted to abstract and solidify the essential salient characteristics of display system operation. The model in and of itself is not particularly useful, however, unless some helpful and pertinent results can be derived from it. In Section 2. 7 a closed-form solution for average response time with no queueing was developed. In this chapter two ways to find the average response time when queueing develops so that there is competition for the use of system resources will be discussed and compared.

3. 1 Simulation

> A simulation of a system or an organism is the operation of a model or simulator which is a representation of the system or organism. The model is amenable to manipulations which would be impossible, too expensive or impractical to perform on the entity it portrays. The operation of the model can be studied and, from it, properties concerning the behavior of the actual system or its subsystem can be inferred. [49 p. 909]

With this broad definition as a base, the definition of simulation can be further narrowed to the field of computer-based Monte Carlo analysis [10, Sec. 5. 4]. A model, manifested as a computer program, is designed to act like a real system. The probabilistic, or stochastic, nature of the real system is reflected by use in the model of a random number generator. By conducting many trials, or simulations, statistics can be gathered concerning operation of the system.

Clearly a common question which must arise in simulation concerns the number of trials needed to give reliable results. If enough trials are performed, Bernoulli's law of large numbers can be invoked to guarantee that the simulation results are convergent to values truly representing characteristics of the model.

The computer program needed to simulate a real system can be prepared in two ways. An assembler or compiler language can be used to write a program tailored to the particular system at hand. This can be time consuming, and can result in an inflexible program in which subsequent system modifications are difficult to represent. However, the program can be tailored to the system being simulated, and thus should take less computer time for execution than a program prepared by other means.

The second way to prepare a simulation program is by writing it in a simulation language. Such a language can resemble either assembly or compiler code. A variety of general purpose simulation languages are reviewed in a reference [55].

While these languages are not necessarily easy to learn, once mastered they are easy to use, and can readily be made to reflect changes in the real system. This flexibility is obtained at the expense of greater computer time requirements than for special programs written to simulate a specific system.

3. 2 Markov Analysis

Markovian analysis techniques, of which queueing theory is an important portion, are based on a well-developed mathematical framework. Of specific interest here is the mathematical entity called a continuous parameter Markov chain, which is described by Parzen [44].

In order that Markovian analysis be applied to the display system model, several restrictions must be applied. First, all service time distributions (for the computer, data link, etc.) must be derivable from the simple negative exponential distribution. This then includes the hyper-exponential distribution. Second, the various branches in the flow of a job through the model must be strictly random (probabilistic), made without regard to where the job has been, or how many times the job has made the branch previously.

If these restrictions are satisfied (how well they are or are not met is discussed later), the system's response time can be found using Markov analysis. This is accomplished by using the Recursive Queue Analyzer (RQA), a computer algorithm developed and implemented by the Systems Engineering Laboratory. [59, 60]

3. 3 Qualitative Comparison of Simulation and Markov Analysis

Both simulation and Markov Analysis have good and bad features. Simulation and simulation languages can be very general, in terms of the allowable structure of a system, the service time distributions which can be specified, and the types of branches which can be modeled. As mentioned in the previous section, Markov analysis is very restrictive of the permissible branches and service time distributions which can be modeled.

Offsetting these restrictions, however, is the short computing time needed by RQA to analyze a queueing system. Specifically, when compared to GPSS (General Purpose Simulation System), the simulation language available at The University of Michigan, RQA needs between only 6 and 18% as much computer time to analyze the same system. The time factor is very important when conducting parameterized studies or performing an optimization (as will be done in this case), because the system model is analyzed not once, but many times.

In addition, Markov analysis need not be concerned with the period and correlation coefficients of random number generation techniques, as is simulation [47, pp. 109-111]. Indeed, simulation results can fail to converge to the correct point as a consequence of an inappropriate random number generator, whereas Markov analysis and its embodiment in RQA are theoretically guaranteed to converge uniquely under certain non-restrictive conditions [60, pp. 6-7], and these results will be correct whenever the modeled system does in fact satisfy the assumptions presented in the previous section.

3.4 Quantitative Comparison of Simulation and Markov Analysis

What if a system is analyzed with RQA even though the necessary assumptions are not satisfied? Are the results still usable, or must simulation be employed? Certainly the computational efficiency of RQA is an excellent motivation for attempting to extend its usefulness in the manner suggested by the preceding two questions.

How is the appropriateness of RQA analysis for non-Markovian systems investigated? There are unfortunately no useful theoretical tools available: the alternative, but less satisfactory course of empirical investigation must be taken. This course is less satisfactory because no general statements can be made concerning when RQA can and cannot appropriately be used. Only statements about specific instances can be made. Thus it is necessary to analyze the display system model with RQA and GPSS for a set of typical parameter values, and carefully study the results.

Tables 3-1, 3-2, and 3-3 present the results of many such RQA and GPSS analyses. For each analysis the long term average service rates and branching probabilities are the same; the corresponding probability distributions, however, are not the same. That is, the display system model has been analyzed several times with GPSS. Each analysis has used different service time distributions and branching distributions; only their averages have remained unchanged. In most cases the distributions do not satisfy the restrictions needed for a Markov analysis with RQA. Despite this, the averages for the various distributions have been used for a Markov analysis of the same model, but with the required negative exponential service time distributions and probabilistic branches.

Number of Display Consoles	Type of Analysis	Server Distribution	Branch Type	Response Time, msec.	Percentage Deviation from RQA Analysis
1	GPSS	Deterministic	Probabilistic	878	-1.6
1	GPSS	Deterministic	Deterministic	854	-4.3
1	GPSS	Uniform	Probabilistic	884	-0.9
1	GPSS	Uniform	Deterministic	885	-0.8
1	GPSS	Exponential	Probabilistic	857	-3.9
1	GPSS	Exponential	Deterministic	854	-4.3
1	RQA	Exponential	Probabilistic	892	0.0
1	Calculated from TL(R, $\underline{X}$, $\underline{Y}$)			892	0.0

Table 3-1

Analysis Results for One User (R=1)

Number of Display Consoles	Type of Analysis	Server Distribution	Branch Type	Response Time, msec.	Percentage Deviation from RQA Analysis
2	GPSS	Deterministic	Probabilistic	996	-3.9
2	GPSS	Deterministic	Deterministic	1001	-3.4
2	GPSS	Uniform	Probabilistic	1019	-1.6
2	GPSS	Uniform	Deterministic	1019	-1.6
2	GPSS	Exponential	Probabilistic	1028	-0.7
2	GPSS	Exponential	Deterministic	1034	-0.1
2	RQA	Exponential	Probabilistic	1035	0.0

Table 3-2

Analysis Results for Two Users (R = 2)

Number of Display Consoles	Type of Analysis	Server Distribution	Branch Type	Response Time, msec.	Percentage Deviation from RQA Analysis
3	GPSS	Deterministic	Probabilistic	1138	-6. 1
3	GPSS	Deterministic	Deterministic	1147	-5. 3
3	GPSS	Uniform	Probabilistic	1179	-2. 7
3	GPSS	Uniform	Deterministic	1187	-2. 1
3	GPSS	Exponential	Probabilistic	1229	+1. 4
3	GPSS	Exponential	Deterministic	1223	+0. 9
3	GPSS	Hyperexponential	Probabilistic	1264	+4. 3
3	GPSS	Hyperexponential	Deterministic	1288	+6. 3
3	RQA	Exponential	Probabilistic	1212	0. 0

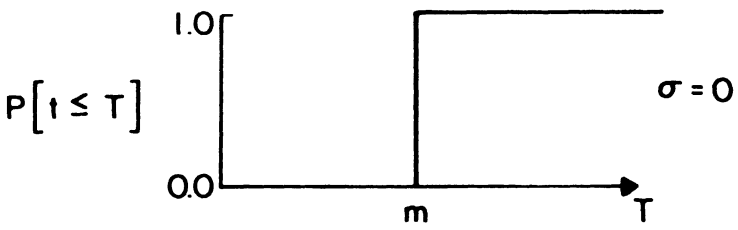
Table 3-3

Analysis Results for Three Users (R = 3)

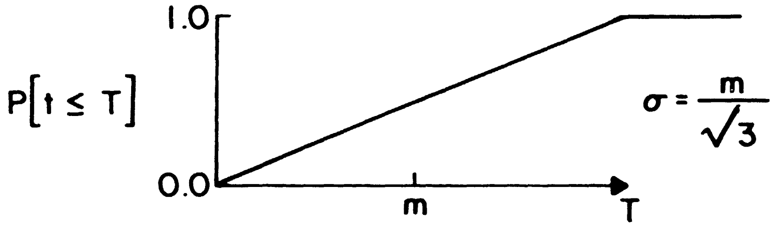
Figure 3-1 illustrates the four types of service time distributions used, and Figure 3-2 illustrates the two distributions used for branching processes. The branches following completion of a remote computer and main computer processing intervals are the only branches to which the deterministic distribution was applied. Also given in Figures 3-1 and 3-2 are the means (m) and standard deviations (σ) for each distribution. Note that σ varies from 0 (deterministic) to 2m (hyper-exponential) for the various service time distributions, thus covering a wide range of variability.

The important results are in the column headed "Percentage Deviation from RQA Analysis" of the three tables. Here can be seen that for exactly the same system, with exponential servers and probabilistic branches, the difference between RQA and GPSS analysis varies from -3.9% to +1.4%. This is essentially an error built into the analyses techniques by factors such as RQA's convergence error [60, pp. 7-8] and GPSS's linear interpolation and truncation vis-a-vis continuous functions [27, p. 28]. Very little of the GPSS error can be attributed to an insufficient number of samples: Table 3-5 shows that after 10,000 interactions with the display system had been simulated, various statistics obtained from the simulation converged nicely.

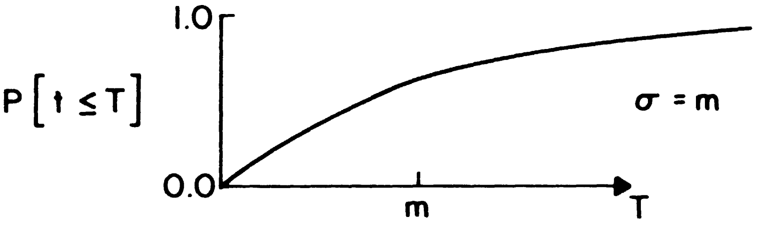
As can be seen from the first three tables, the deviations for all cases are quite small, the worst being 6.3% It is thus very reasonable to conclude that for this model, and for parameter values of the order used in the model and given in Table 3-4, it is appropriate



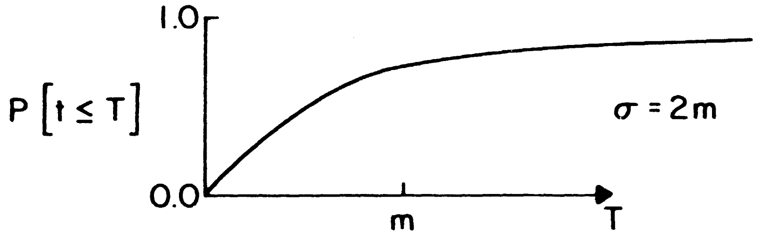
(a) Deterministic Distribution



(b) Uniform Distribution



(c) Negative Exponential Distribution

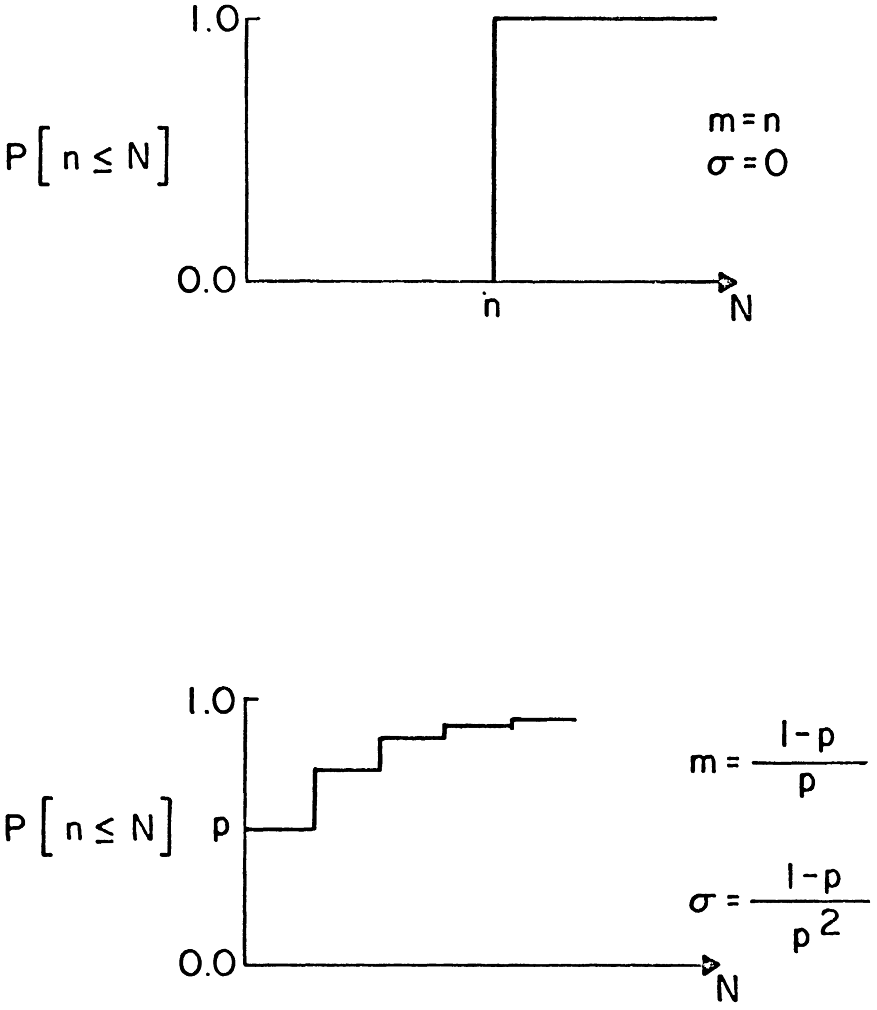


(d) Hyperexponential Distribution

t = Service Time

Figure 3-1

Service Time Distributions



p= Probability of Not Accessing Bulk Storage

n= Number of Accesses to Bulk Storage Before Service Complete

Figure 3-2

Branching Distributions

t_1 = 150 msec.

t_2 = 300 msec.

t_3 = 100 msec.

t_4 = 90 msec.

t_5 = 100 msec.

t_6 = 50 msec.

t_7 = 100 msec.

t_8 = 400 msec.

t_9 = 90 msec.

t_{10} = 100 msec.

t_{11} = 50 msec.

PM = .5

PDRC = .5

PRD = .5

PD = .5

Table 3-4

System Parameters Used for RQA and GPSS Analysis

Number of Simulated Interactions	Response Time, msec.	Total Throughput Time for Server 1, msec.	Total Throughput Time for Server 6, msec.	Utilization factor for Server 8, %
1000	796	160	69	39
2000	776	161	70	38
4000	784	163	68	38
6000	786	163	67	38
8000	786	163	66	38
10000	788	163	66	39

Table 3-5

Convergence of Simulation Results

to substitute a short inexpensive RQA analysis for a longer, more expensive GPSS analysis, even though the model being analyzed may not always satisfy the conditions for RQA analysis. Similar conclusions for another mathematical model are reported in a reference [20]. These differences in average response time are small because of the relatively low level of congestion and queueing in the model, even with three users (Table 3-3). With little queueing, the response time just is not sensitive to changes in second order statistics of the service time and branching distributions. If there were more queueing, the differences in response times would be larger, so that use of RQA might not then be justified unless it were known that the various distributions satisfied the Markov requirements. However, in the work which follows, it is possible to use only RQA for analysis.

Chapter IV

OPTIMIZATION PROCEDURE

The goal of this chapter will be to develop an optimization procedure which, together with the previously introduced display system model, can determine which one of many display systems is best for a particular application. The model, when analyzed with the Recursive Queue Analyzer (RQA) as discussed in Chapter III, gives a prediction of response time for a specific display system. The optimization procedure, by judicious use of RQA, will find optimum display systems. An optimum display system is one which minimizes response time, subject only to a dollar cost constraint. In Section 4. 1 the exact nature of the optimization is examined, while Section 4. 2 presents the actual optimization algorithm.

4. 1 Problem Formulation

The optimization problem being confronted here is one in four dimensions, corresponding to the four subsystems of a display system, namely the data link, the remote computer's bulk storage, its core storage, and the remote computer-display control. Associated with each of these subsystems is a variable. These are denoted X1, X2, X3, and X4, as defined in Section 2. 2, and are collectively designated as $\underline{X} = (X1, X2, X3, X4)$. Four functions of $\underline{X}$ have been defined. They are $C'(\underline{X})$ (equation 2. 10), the monthly hardware and main computer storage costs for a display system; $C(\underline{X})$ (equation

2.9), the total monthly costs of a display system; $TL(R, \underline{X}, \underline{Z})$ (equation 2.14), the lower bound on system response time; and $T(R, \underline{X}, \underline{Z})$, (equation 2.11), the system's actual response time. Also, R is the number of active display consoles attached to a remote computer, $\underline{X}$ is a vector describing the system's hardware, and $\underline{Z}$ is a vector describing the system's application. In Section 2.6, $C'(\underline{X})$ was shown to be monotonic nondecreasing with respect to each component of $\underline{X}$ under very mild restrictions. $T(R, \underline{X}, \underline{Z})$ was demonstrated in Section 2.7.2 to be monotonic nonincreasing, also under reasonable restrictions.

Thus the situation is such that decreasing response time increases cost, and vice-versa. This is clearly the basis for an optimization. The manner in which the optimization is approached must be very strongly influenced by the amount of computer time needed to calculate a display system's response time. That is, when more than one display console is serviced by a remote computer, the Recursive Queue Analyzer (RQA) must be used to find response time. While RQA is faster than GPSS, as discussed in Section 3.3, it is still time consuming. A single analysis of the display system model can take as much as 30 seconds of CPU time and cost $3.00. It is therefore necessary to devise an optimization algorithm which minimizes the number of response time evaluations, without compromising in any way the final solution's accuracy.

While seeking out a convenient way to handle the optimization, it is necessary to determine if it should be performed in the domain of discrete or continuous variables. A bit of thought reveals several very persuasive arguments favoring discrete variables. First, the optimization is meant to deal with real currently available subsystems. There is not available a continuous spectrum of data link speeds, memory sizes, or computing powers. Rather, only very specific capacities can be procured, and this will be just as true in the future as it is now. A second consideration is the nonlinearity (Sections 2. 6 and 2. 7) of the functions being dealt with. If they were linear, the problem could be solved using continuous variables and linear programming, and then rounding up or down elements of the solution in various combinations to find an optimum. But because of the nonlinearities there is absolutely no guarantee that this method, used here, would yield an optimum solution. Thus it seems best to approach the discrete optimization head-on.

This can be done in one of three ways. The first is to minimize cost while imposing a constraint on response time, the second is to minimize response while constraining cost, and the third is to minimize some function of both cost and response time. Of these three, only the second can be considered as satisfactory because it permits an implementation which minimizes the number of times RQA must be used, if the cost of $C'(\underline{X})$ rather than $C(\underline{X})$

as a cost constraint is satisfactory. This must be accepted, because determining $C(\underline{X})$ first requires use of RQA, which is not realistic to do, as $C(\underline{X})$ would be evaluated quite often, and using RQA just once takes from 10 to 30 seconds of CPU time. In any event this issue is not of particular importance, because experience indicates that the difference between $C(\underline{X})$ and $C'(\underline{X})$ is quite small.

4. 2 Optimization Algorithm

With this background material in mind, an appropriate optimization algorithm will be presented, the first part of which is an adaptation of work done by Lawler and Bell [41]. Recalling that $\underline{X} = (X1, X2, X3, X4)$, let each Xi be bounded such that $0 \leq Xi \leq Xi_{max}$, where Xi_{max} is, in turn, exactly one less than a power of two. It is not necessary that all Xi_{max} be equal. This bound is quite reasonable because in practice there are only a finite number of each subsystem available.

Now let X be the binary vector formed by concatenating the binary representation of the Xi's, including leading zeros, if any. If $Xi_{max} = 15$ for all i, and $X = (5, 12, 9, 15)$, then $X=(0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1)$. Eliminating the commas, this is written $X = (0101110010011111)$. Four definitions are now made.

Definition 1: Given $\underline{X}$ and $\underline{Y}$, any four component vectors such that $0 \leq Xi \leq Xi_{max}$, and also given X and Y, formed from $\underline{X}$ and $\underline{Y}$ as prescribed above. If each bit of X is less than or equal to each bit of Y, then $X \leq Y$. This defines a "vector partial ordering" of X and Y.

Example: $(1011011) \leq (1111011)$, and $(0111011) \not\leq (1011011)$.

Definition 2: If $V = (v_1, v_2, \ldots, v_n)$ is a binary vector, then the numerical value of V, n(V), is $v_1 2^{n-1} + v_2 2^{n-2} + \ldots + v_n 2^0$.

Example: If V = (1011001), $n(V) = 64 + 0 + 16 + 8 + 0 + 0 + 1 = 89$.

Definition 3: If the numerical value of X is less than or equal to the numerical value of Y, $n(X) \leq n(Y)$. This defines a "numerical ordering" of X and Y.

Example: $n(111001) \leq n(111100)$, and $n(1100101) \not\leq n(0111010)$.

Note: $X \leq Y \implies n(X) \leq n(Y)$.

Definition 4: If Y is such that $n(X) < n(Y)$ but $X \not\leq Y$, and there exists no other vector, say Y', for which $n(Y') \leq n(Y)$ and $X \leq Y'$, then Y will be called X*. That is, X* is the first vector numerically larger than X which is not also larger in the vector partial ordering sense.

Example: If X = (0101100), X* = (0110000). If X = (0101011), X* = (0101100).

This X* can be readily computed by taking the bit-by-bit logical 'or' of X with X-1, and then adding 1, with the exception that if $X = (00 \ldots 0)$, $X^* = (00 \ldots 1)$. Addition and subtraction on the vector X is handled by performing the operation on n(X), and converting back to a vector.

Lemma 1: By definition 4, X* is the first vector numerically larger than X which is not also larger in the vector partial ordering sense. Therefore X* - 1 is larger than X in the vector partial ordering sense.

Just as X is derived from $\underline{X}$, there is also an $\underline{X}^*$ from which X* derives. The same is true for X* - 1. Having said this, the following theorem, adapted from reference 28, can now be stated and used.

Theorem 1: If $\underline{X} = (X1, X2, X3, X4)$ and $X^*-1 = (X1', X2', X3', X4')$ then $Xi \leq Xi'$ for $i = 1, 2, 3, 4$.

Proof: By Lemma 1, it is known that $X \leq X^*-1$ in the vector partial ordering, and therefore that each bit of X is less than or equal to the corresponding bit of X*-1. Now each Xi and Xi' is represented in binary form as a sequence of corresponding bits in X and X*-1, respectively. Let these sequences be denoted as u_i and v_i, respectively. Then each bit of u_i is less than or equal to the corresponding bit of v_i, so that $n(u_i) \leq n(v_i)$. But $Xi = n(u_i)$ and $Xi' = n(v_i)$, so that $Xi \leq Xi'$, for $i=1,2,3,4$.

Associated with Theorem 1 is

Corollary 1: $C'(\underline{X}) \leq C'(\underline{X^*-1})$, and $T(R, \underline{X}, \underline{Z}) \geq T(R, \underline{X^*-1}, \underline{Z})$.

Proof: $C'(\underline{X})$ and $T(R, \underline{X}, \underline{Z})$ are monotonic nondecreasing and nonincreasing, respectively, with respect to each component of $\underline{X}$. From Theorem 1 it is known that each component of $\underline{X}$ is less than or equal to the corresponding component of $\underline{X^*-1}$.

This corollary is useful in the algorithm of Figure 4-1. With C'_{max} an upper limit on $C'(\underline{X})$, the purpose of this algorithm, which is just a portion of the entire optimization, is to find a set S of vectors, such that if $C'(\underline{X^*-1}) \leq C'_{max}$, then $\underline{X^*-1} \in S$; if not, but $C'(\underline{X}) \leq C'_{max}$, then $\underline{X} \in S$. Thus, every vector in S is feasible (that is, for $Y \in S$, $C'(\underline{Y}) \leq C'_{max}$), but not every feasible vector is in S. For instance, if X= (011000000000) and X*-1=(011111111111) and X*-1 is feasible, none of the hundreds of vectors (all of which are feasible) in the vector partial ordering between X and X*-1 will be in S; only X*-1 itself will be. This can be done because T(R, (011111111111), $\underline{Z}$) is known to be no greater than the response time obtained from display systems represented by any of these intermediate vectors, and it will, in fact, usually by less.

Thus it is seen that the algorithm in Figure 4. 1 depends for its success on the number of vectors falling between X and X*-1 when X*-1 is feasible. The more intermediate vectors there are,

Figure 4-1

Formation of the Set S

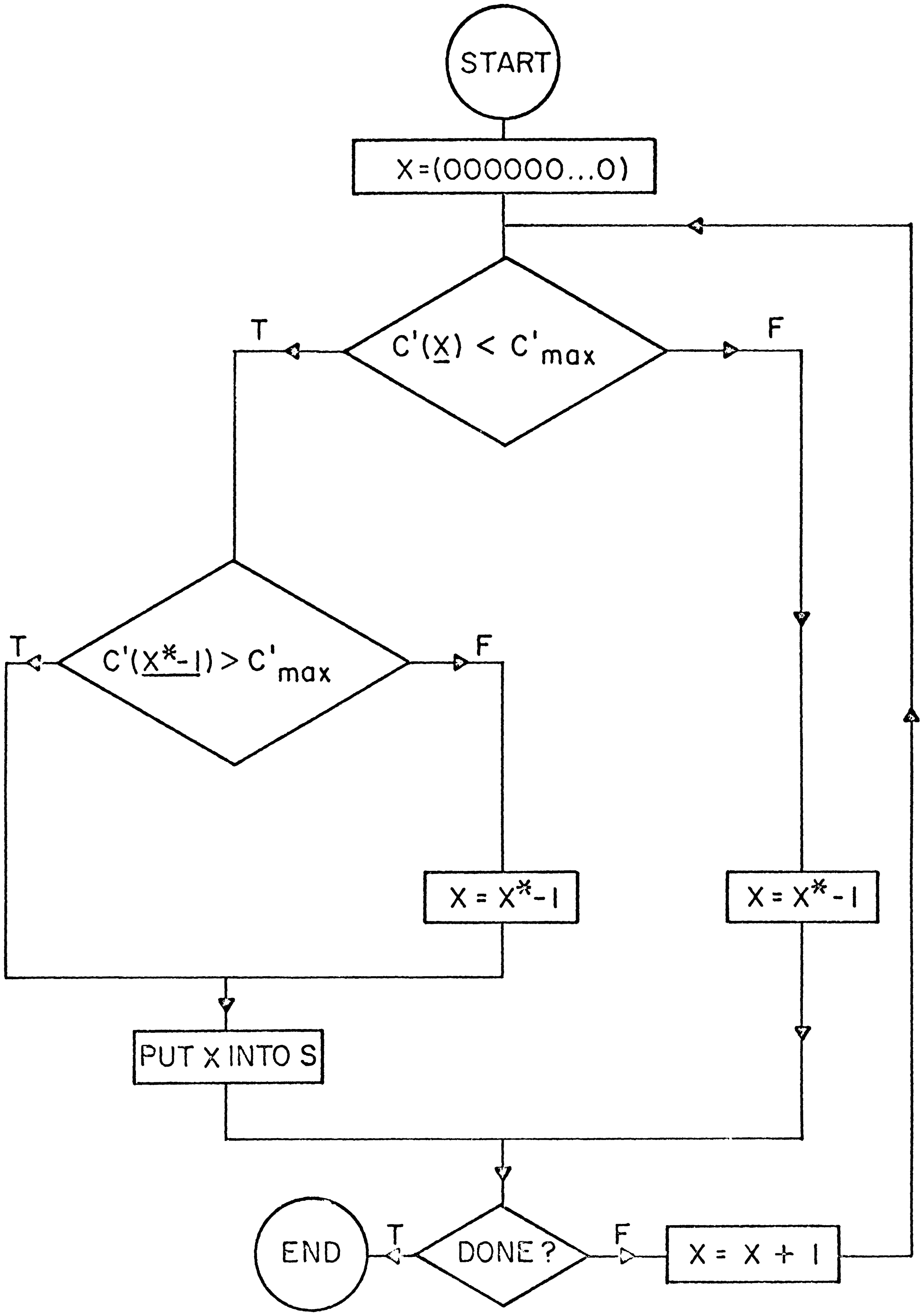

START
X=(000000...0)
C'(X) < C'max
T
F
C'(X*-1) > C'max
T
F
X = X*-1
X = X*-1
PUT X INTO S
DONE ?
T
F
END
X = X + 1

Figure 4-1

the smaller will S be. However, if X*-1 is not feasible, the interval between X and X*-1 must be reduced to smaller intervals and searched. The definition of X* used here gives a reasonable balance between successfully eliminating large intervals of vectors and efficiently conducting a finer search over intervals which cannot be eliminated.

There is another definition of X* which would be equally suitable here, and which might provide an even more efficient search. It is obtained by redefining vector partial ordering and numerical value. Define $\underline{X} < \underline{Y}$ if $Xi \leq Yi$, $i = 1, 2, . . . , 4$. This is the vector partial ordering. Also, define $n(\underline{X}) = X1(M^3) + X2(M^2) + X3(M^1) + X4(M^0)$, with $M = \max \{Xi\}$. This is the numerical value. Now with these redefinitions, Definition 4 for X* can be used, with the result that if:

$$X = (0, 1, 0, 0), \quad X^* = (1, X2_{max}, X3_{max}, X4_{max})$$

$$X = (3, 6, 5, 1), \quad X^* = (3, 6, 5, X4_{max})$$

$$X = (3, 6, 5, 0), \quad X^* = (3, 7, X3_{max}, X4_{max}).$$

The usefulness of this alternate definition has not been explored, because no matter what definition of X* might be used, the following step in the optimization will yield the same result.

Now, if nothing else were known about the problem, the optimum solution could be found by an exhaustive search with RQA over the set S, whose elements are feasible solutions. This in itself is far better than searching over *all* feasible solutions with RQA. However, further improvements can be made. Specifically, the next theorem further exploits the monotonicity of T.

Theorem 2 : If, for $\underline{X}$, $\underline{V} \in S$, $Xi \leq Vi$ for i=1, 2, 3, 4, then $T(R, \underline{X}, \underline{Z}) > T(R, \underline{V}, \underline{Z})$.

Proof : In Section 2. 7. 2 it was shown that $T(R, \underline{X}, \underline{Z})$ is monotonic nonincreasing with each element of $\underline{X}$. Then, since each component of $\underline{X}$ is less than or equal to the corresponding component of $\underline{V}$, the monotonic property of T means that $T(R, \underline{X}, \underline{Z}) > T(R, \underline{V}, \underline{Z})$.

Theorem 2 results in

Corollary 3 : If for $\underline{X}$, $\underline{V} \in S$, $Xi \leq Vi$ for $i = 1, 2, 3, 4$, then $\underline{X}$ can be deleted from S.

Proof: By Theorem 2, $T(R, \underline{X}, \underline{Z}) \geq T(R, \underline{V}, \underline{Z})$. Thus there is no reason to keep X in S, because the goal is to minimize T.

By systematically comparing each vector in S to every other vector in S, and by eliminating vectors whenever appropriate, a new set S' can be formed, such that $N(S') \leq N(S)$, where $N(\cdot)$ represents the cardinality of a set. The number of comparisons needed lies between $\frac{[N(S)][N(S) - 1]}{2}$ and $\frac{[N(S') - 1][N(S') - 2]}{2} + [N(S) - 1]$. Given an arbitrary ordering of S, the upper limit is found by assuming that no vectors can be eliminated from S. Then the first vector is compared to N (S) - 1 vectors, the second to N(S)-2, etc. This is just the finite sum $[N(S)-1] + [N(S)-2] + \dots + [1] = \frac{[N(S)][N(S)-1]}{2}$. For the lower limit, assume that comparing the first vector to the N(S)-1 other vectors results immediately in reduction in cardinality to N(S'). Then the second vector in S' is compared to N(S')-2 vectors, the third to N(S')-3, etc., for a total of $[N(S)-1] + [N(S')-2] + [N(S')-3] + \dots + 1 = [N(S)-1] + \frac{[N(S')-1][N(S')-2]}{2}$.

Once again, if nothing more were known about the problem, an exhaustive search of S' with RQA would be necessary. This could be a very time-consuming process. Still further improvements are possible. The algorithm of Figure 4-2 can now be utilized to find the vector in S' which minimizes $TL(R, \underline{X}, \underline{Z})$. $TL(R, \underline{X}, \underline{Z})$ is easily calculated, as RQA is not needed. Call this vector $\underline{A}$. Unfortunately, $\underline{A}$ does not necessarily also minimize $T(R, \underline{X}, \underline{Z})$, unless R=1, in which case the optimization is complete. However, experience indicates that it often comes close to doing so.

The optimization is completed with the algorithm in Figure 4-3. TMIN is set equal to $T(R, \underline{A}, \underline{Z})$. This gives a good upper bound on the true TMIN. RQA need be used only when $TL(R, \underline{X}, \underline{Z}) < TMIN$. If it happens that $T(R, \underline{X}, \underline{Z}) < TMIN$, then TMIN becomes $T(R, \underline{X}, \underline{Z})$ and $\underline{A}$ is replaced by $\underline{X}$. Note that whenever T or TL is calculated, the interaction assignment procedure of Section 2. 7. 1 must first be performed. When S' has been

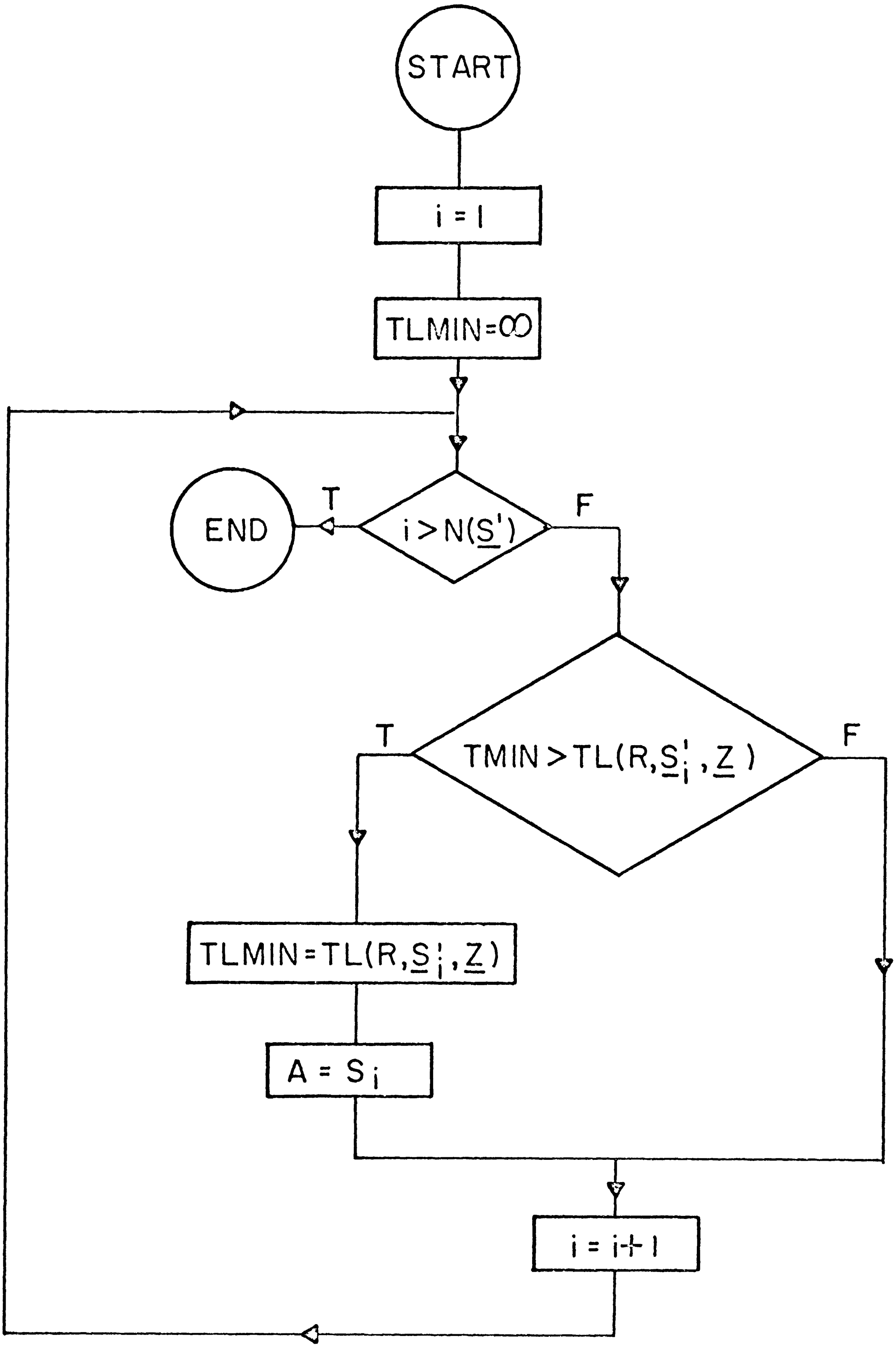


Figure 4-2
Minimization for One User (R=1)

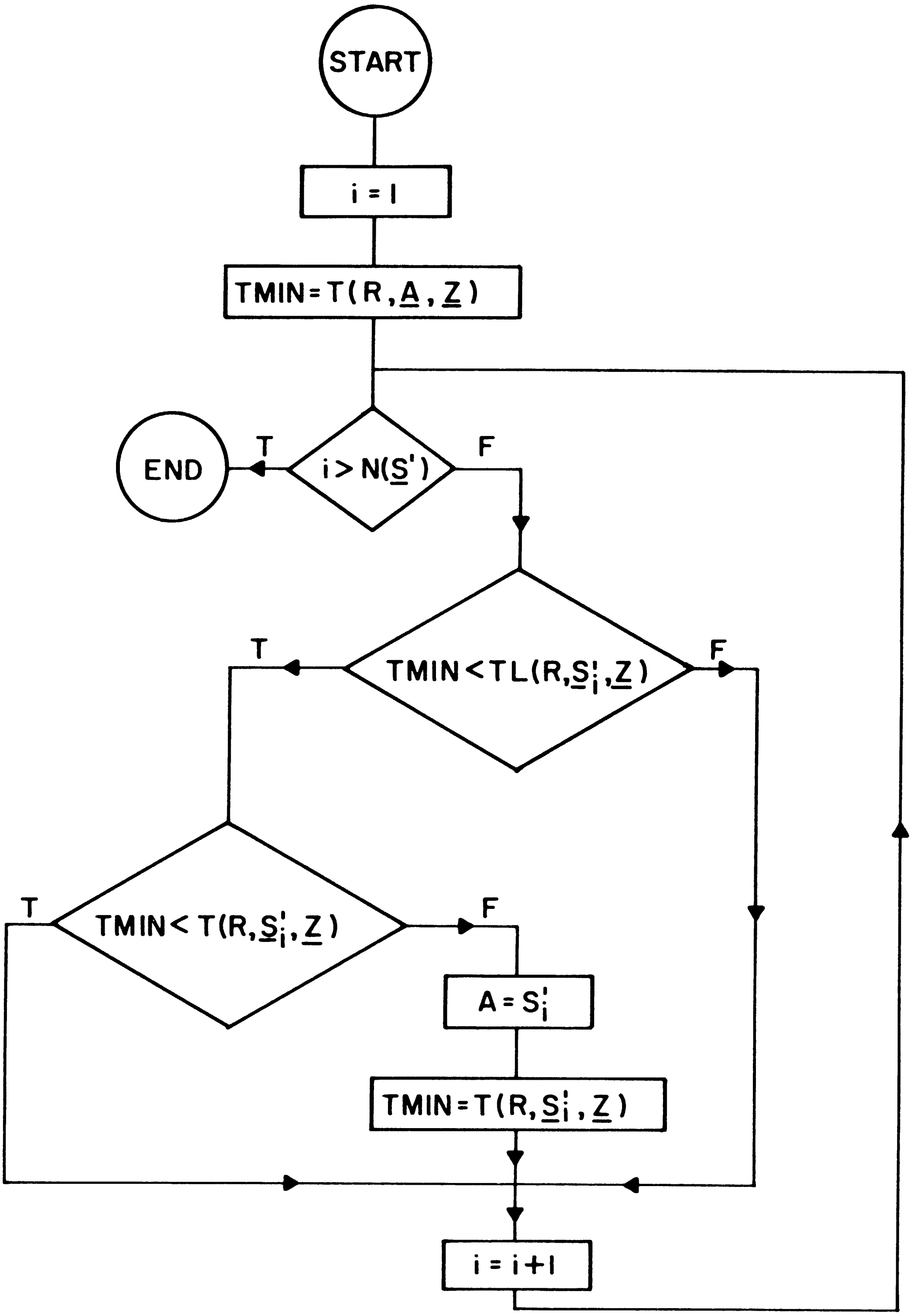


Figure 4-3

Final Minimization

exhaustively searched, $\underline{A}$ represents the display system which minimizes response time for a cost $C'(\underline{A}) \leq C'_{max}$. Appendix B presents a complete discussion of the optimization programs.

In this procedure steps have been taken to make the initial value of TMIN as small as possible. This is because the smaller TMIN is, the less often will the use of RQA be called for. The statistics in Table 4-1 are indicative of the very real success achieved in using RQA as infrequently as possible. Note that the number of RQA runs is less than N(S'), which is, in turn, less than N(S), as is desired.

Total Number of Systems	N(S)	N(S')	RQA Runs
1280	6	4	1
1280	704	26	3
1280	931	12	1
1280	159	5	2
1280	800	23	4
1280	874	13	3
960	3	3	1
960	190	17	2
960	706	6	1
640	2	2	1
640	138	16	2
640	379	14	1
640	156	16	4

Table 4-1

Optimization Statistics

Chapter V

EVALUATION OF COMPUTING POWER

A critical necessity in the optimization discussed in Chapter 4 is the creation of a suitable data base of hardware subsystems from which an optimal display system can be chosen. This can rather easily be done for all subsystems except the remote computer-display control subsystem for which not just computing power, but also display capability must be determined. Display capability must be known because not every display control can display the same amount of information. Only those able to display more than some minimum amount of information can even be considered for inclusion in a display system; this minimum amount of information is application dependent. Having eliminated unacceptable display controls, the remaining display control-remote computers must be rated according to their computational capabilities. In Section 5.1 a method of evaluating the display capability of various display controls will be discussed. Section 5.2 reviews past approaches to measuring computing power, while the next section builds on these approaches to develop a technique for measuring a remote computer-display control's power. Section 5.4 combines the results of the preceeding sections, applying both computing power and display capability criteria to remote computer-display controls.

5.1 Displayable Information

Differing technologies, techniques, and design criteria have resulted in display controls with a wide range of display and computational capabilities. The intent in this section is to present a means of measuring display capabilities for comparative evaluation purposes. The evaluation can be performed in the manner employed by Adams Associates in The Computer Display Review [2]. The exact method is to define five different display test patterns which are intended to be representative of as many different applications, specifically, alphanumeric work, weather mapping, mathematical graphing, architectual drawing, and circuit analysis. A display control is rated in terms of the percentage of each pattern which can be presented flicker free. This will be denoted Q_i, i=1, 2, 3, 4, 5. The resulting figures (some or all can be greater than 100%) represent a quantitative rating of the display control. Figures 5-1 through 5-5 show the actual test patterns.

To obtain a single rather than multiple rating for a display control, a weighted average of the five individual ratings can be taken, with the weights based on the relation of each test pattern to the application for which the equipment is being evaluated. The more typical of an application a test pattern is, the higher the weight. These weights, of course, must sum to unity. They will be denoted by Ω_i, i=1, 2, . . . , 5.

```
                       BURNER SYSTEM

   SYSTEM VALVES       SYSTEM STATUS       SYSTEM SIGNALS

SUPERHEATER        C      *TRIPPED*      STEAM PRESSURE     N
REHEATER           C                     FURNACE PRESSURE   N

MAIN GAS SHUTOFF   0                     GAS TEMPERATURE    N
         VENT      0                     COMBUSTION         M

MAIN OIL SHUTOFF   C                     FORCED DRAFT FAN   R
         RETURN    C                     AIR HEATER         0
         NORTH     C
         SOUTH     C                     IGNITOR CHARGED    Y
                                         GAS READY          Y

IGNITOR SHUTOFF    0
         VENT      C                     OIL READY          N
                                         DUAL PERMISSIVE    N

PENTHOUSE NO. 1    0
          NO. 2    0                     MODE SWITCH        C
          VENT     0                     PURGE REQUIRED     Y

                         BURNERS

              1    2    3    4    5    6    7    8    9    10
STATUS        -    -    S    *    *    *    I    -    -     -

IGNITOR
      FLAME   -    -    -    -    -    -    -    -    -     -
      TRANS   -    -    -    -    -    -    -    -    -     -
    SHUTOFF   C    C    C    C    C    C    C    C    C     C

MAIN FLAME    -    -    N    F    F    F    -    -    -     -

GAS SHUTOFF   C    C    0    0    0    0    C    C    C     C

OIL SHUTOFF   C    C    C    C    C    C    C    C    C     C
       UNIT   0    0    0    0    0    0    0    0    0     0
       ATOM   I    I    I    I    I    I    I    I    I     I
      PURGE   C    C    C    C    C    C    C    C    C     C
       COOL   C    C    C    C    C    C    C    C    C     C

DAMPER        0    0    0    0    0    0    0    0    0     0

CONTROL
    IGNITOR   -    -    -    -    -    -    -    -    -     -
        GAS   -    -    X    X    X    X    X    -    -     -
        OIL   -    -    -    -    -    -    -    -    -     -
       DUAL   -    -    -    -    -    -    -    -    -     -
        OFF   X    X    -    -    -    -    -    X    X     X
```

Figure 5-1

Alphanumeric Test Pattern

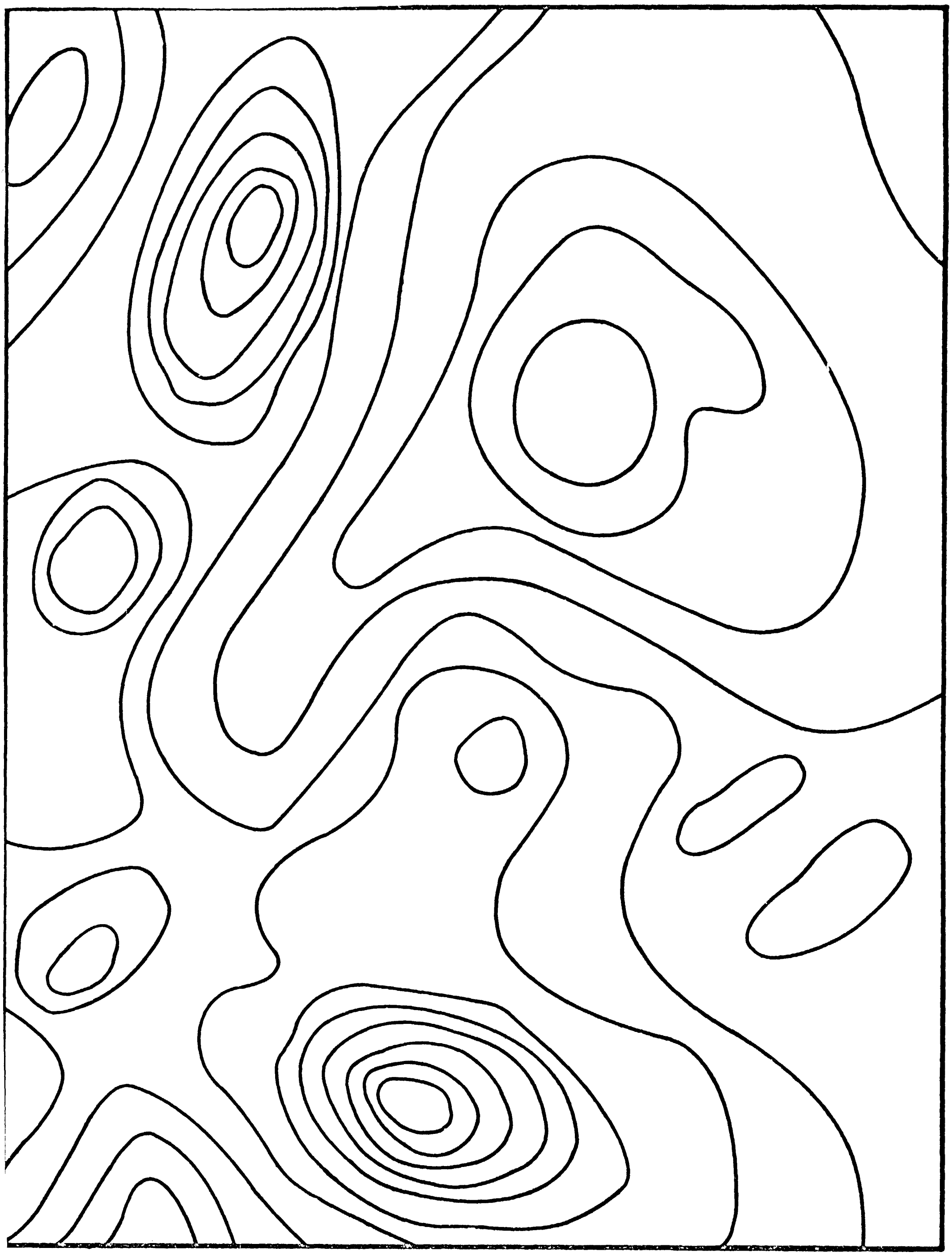


Figure 5-2

Weather Map Test Pattern

Copyright 1967, Adams Associates - Used by Permission

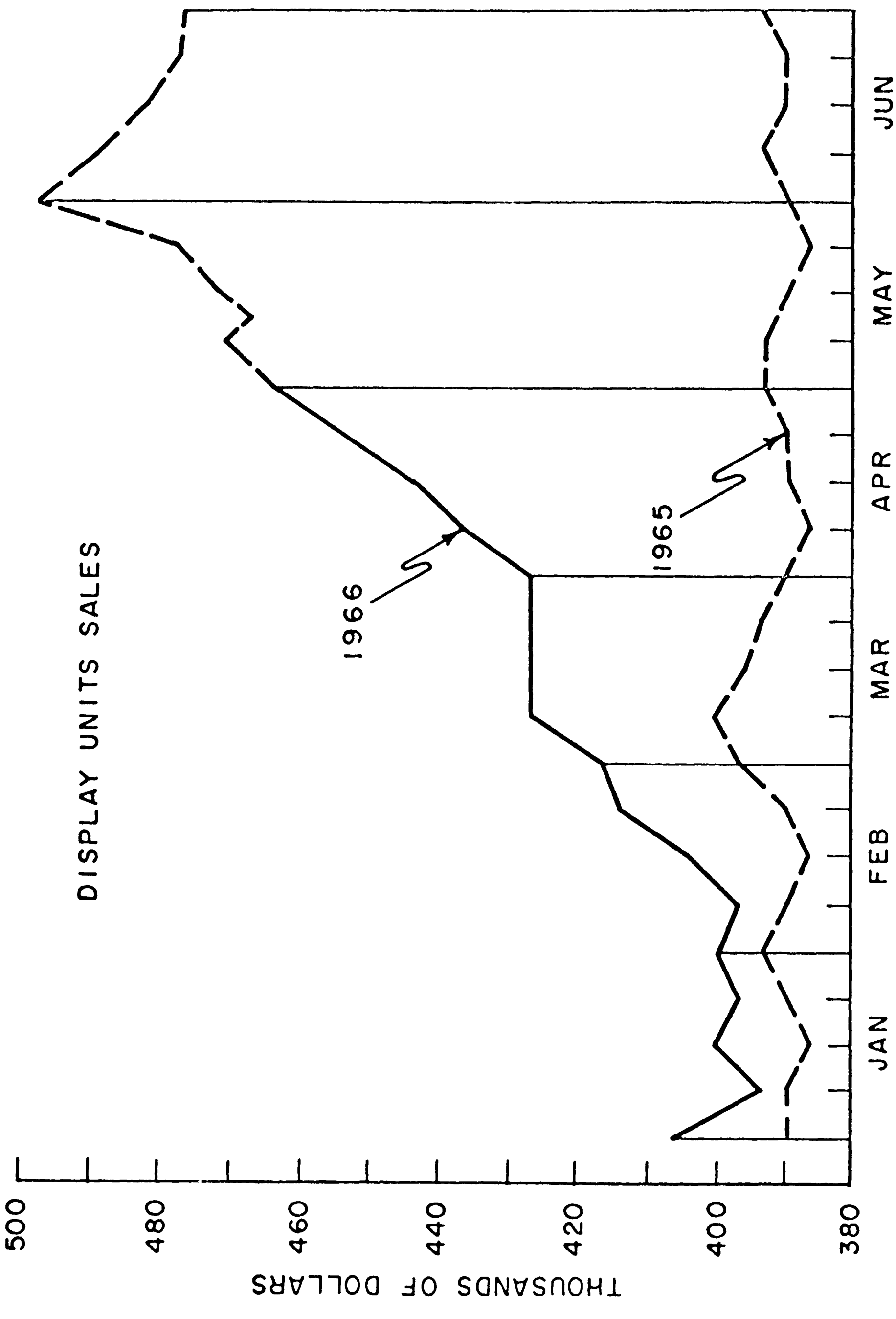


Figure 5-3

Graph Test Pattern

Copyright 1967, Adams Associates - Used by Permission

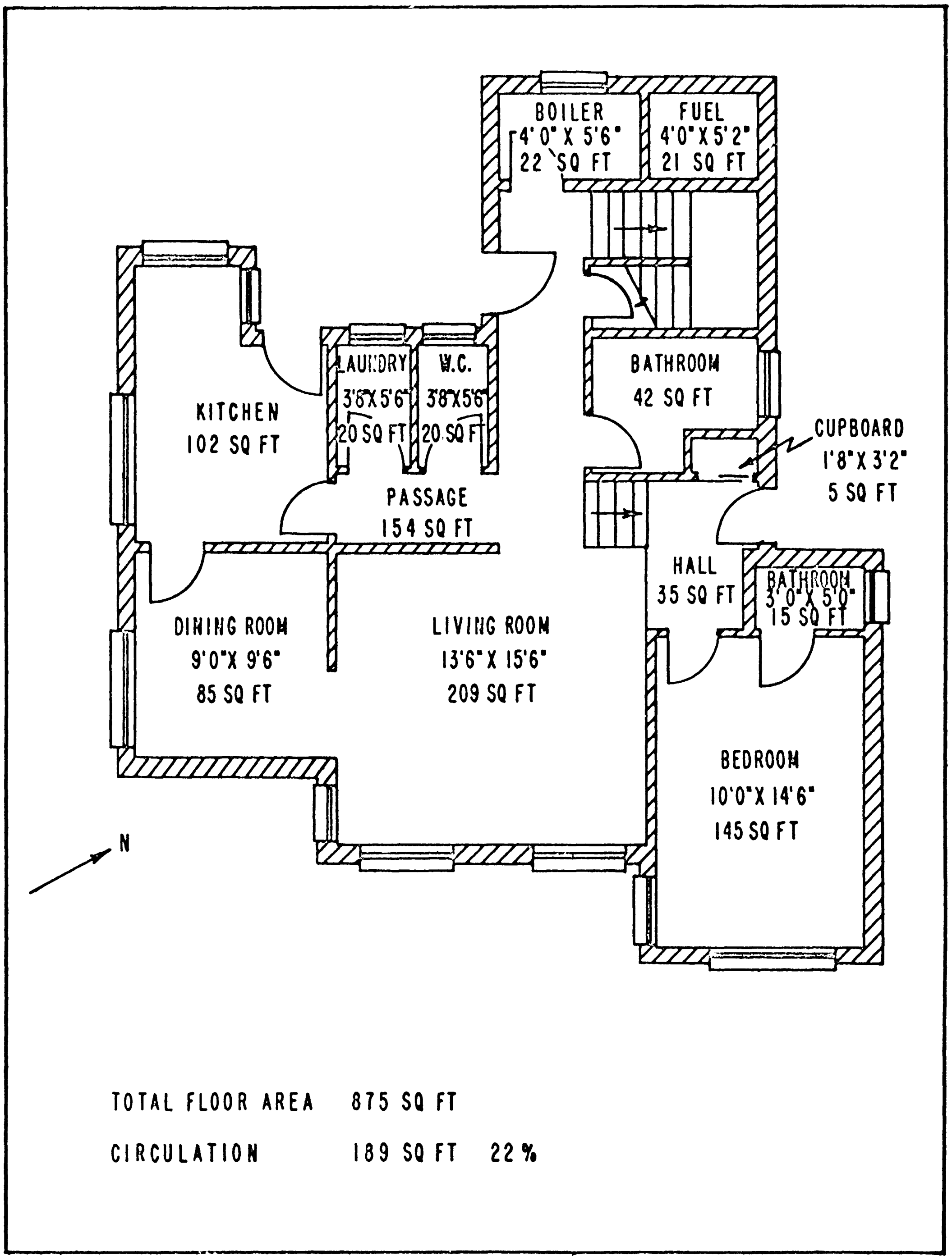


Figure 5-4

Architectural Drawing Test Pattern

Copyright 1967, Adams Associates - Used by Permission

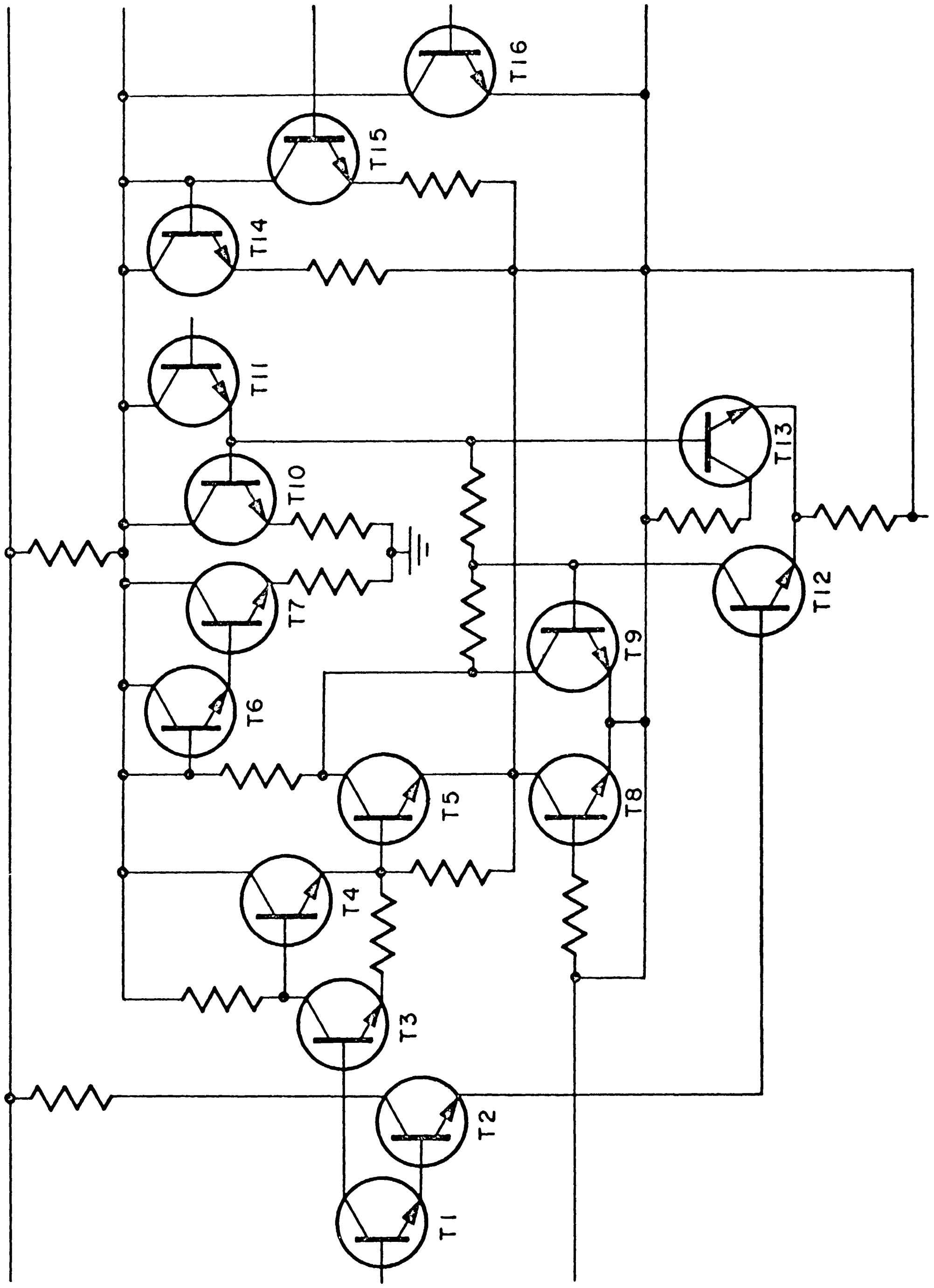


Figure 5-5

Electronic Schematic Test Pattern

Copyright 1967, Adams Associates - Used by Permission

In the case that none of the test patterns is really representative of the application, appropriate patterns can be generated and evaluated. Before taking this step, however, a second look at the test patterns is in order, because some of them may really represent an application quite well without using the application's terminology. For example, the weather mapping pattern, Figure 5-2, has a structure similar to that of both geographical or mathematical contour mapping, as well as sheet metal styling. Likewise, the circuit analysis pattern, Figure 5-5, is not unsimilar to certain mechanical analysis systems.

If unhappily, it does become necessary to create new test patterns, their evaluation in terms of various display controllers is a simple, albeit tedious, task. All that is required is the generation of the list of display commands corresponding to the new test pattern, and a summation of the plotting times for each of the individual display instructions. If this time is Tp milliseconds, and the refresh rate needed for a flicker free display is f per second, then the percentage of the test pattern which can be displayed flicker-free is $Q_{Tp} = 100\% \left(\frac{1000}{f*Tp}\right)$. Because most display controls have similar display commands, virtually the same display file can be used to evaluate many display controls. Q_{Tp} is found here in precisely the same way that the ratings Q_i were found, and is meant to replace the use of the Q_i's when they are not applicable.

This analysis method is easily extended to cases where more than one display console is to be driven by a single display control. If the R display consoles which are to be driven all display the same information, there is no resulting loss of display control capacity on a per display basis. At the opposite end of the spectrum, when all R consoles present different information, each can, on the average, present just 1/R times the quantity of information presented with only one console. In general, for applications where only a fraction x of the information in each display is different, the displayable information per console becomes

$$QR = \frac{Q}{x(R-1)+1} \qquad (5.1)$$

where $Q = \sum_i \Omega_i Q_i$ in the case that the Adams Associates test patterns are used, and $Q = Q_{Tp}$ if a new test pattern has been developed.

5.2 Computing Power - Historical Approaches

Evaluation of the computational power of a display control-remote computer is a more difficult process than the preceding one. A wide range of capabilities exists in display controls, and an evaluation technique must cover all possibilities. For instance, hardware used by the Electronic Systems Laboratory at MIT includes hardware rotation, scaling, and pentracking capabilities [53], while most other equipment includes none of these facilities. Specifically, the evaluation must be sensitive to all hardware features which may be implemented in the display control and remote computer, so that the hardware-software tradeoffs discussed in Chapter 1 can be quantitatively treated.

Furthermore, it is necessary to evaluate the computational speed of the computer-control combination assuming that infinite core memory is available, thereby eliminating I/O delays. This is because the display system model which has been postulated in Chapter 2 treats I/O delays explicitly, rather than lumping them with computing time. It is also desirable, of course, that the evaluation technique be as simple as possible, while still being accurate and taking into account the application to be implemented on the remote display terminal.

Many evaluation methods are available [5]. Perhaps the best known, and most controversial, is Grosch's Law [3, 33, 34, 35, 51], which states that within a group of technologically similar computers, computing power varies as the square of the computer's cost. This law has been proven and disproven several times in the past. Unfortunately, there still remains the necessity of evaluating the power of at least one computer as a basis for the law, and furthermore, there is no evidence that small computer-dispaly controls obey the law. A second problem is separating the cost of the logical portion of a display control from that of the deflection circuitry.

Both Knight [33, 34, 35] and Gruenberger [21] have proposed formulae which attempt to relate a computer's design characteristics to its processing power. The formulae, however, include variables such as the amount of core storage and I/O speed, which are not being considered here. Once again, also, another evaluation technique is required to verify the formulae's applicability.

Benchmark [24, 30] problem testing is often the best evaluation method, because it involves actually running typical programs on real hardware. This is fine for existing systems with existing software, but virtually impossible for evaluation of proposed systems. The quandary of defining a typical problem can be very serious in scientific and graphical applications, so that this method must be discarded for these multiple reasons.

Machine language instruction mixes provide an accurate evaluation technique, and are well-suited to comparing machines of similar design [5]. Difficulties arise when comparing machines with different word lengths, indexing schemes, and number of accumulators. This method is based on determining the frequency with which the various machine instructions are executed, and calculating the time required to execute a fixed number of instructions. Consequently, the machine's application can be accounted for by varying the instruction frequencies. The difficulties mentioned above, however, make this method unsuitable.

Another good technique, kernel problem comparisons, involves calculating the execution time for an algorithm deemed typical of the computer's intended application [5]. This means producing an approximation to the machine code required to implement the algorithm, and a subsequent timing analysis which, for the case at hand, ignores I/O delays. The resulting execution time is then useful to the extent that the kernel problem is truly representative of the overall application. As with benchmark problem evaluation, defining a typical problem is easier said than done. The problem should exercise the various machine capabilities with the same frequencies which will occur in practice. While this may be easy for many business applications, it will probably be difficult for most graphical applications.

In the next section a more satisfactory means of evaluating computing power will be proposed.

5.3 Computing Power - Display Instruction Mix

A combination of the machine instruction mix and kernel problem analysis, by bringing together the two methods' good points while minimizing their bad points, seems feasible. This proposed new method will be called the display instruction mix technique. The first step in its implementation is to define an exhaustive set of basic operations, named display instructions, which might be performed during a graphical man-machine interaction. These display instructions are at a level higher than machine code, thereby removing the generality constraints implicit in the instruction mix approach, but still are at a low enough level so that their selection can remain independent of the data structure and executive system and not involve extensive machine language coding. Also, because there are many display instructions, a broad spectrum of display applications are representable with a good display instruction mix.

Table 5-1 is one possible list of display instructions. A display application is characterized by a set of weights $\{w_i\}$ which sum to unity and represent the relative execution frquency for each display instruction.

Now to evaluate the computing power of a particular display control-remote computer, machine instruction sequences are written to implement each display instruction, and their execution times τ_i calculated. Then the hardware's average instruction execution rate, defined as the variable X4 in Chapter 2, is given by

$$X4 = \frac{1}{\sum_i (w_i \ \tau_i)} \qquad (5.2)$$

This method does in fact require that some machine coding be done; most of it, however, will be very straightforward, and of course does not have to be debugged. What is needed, clearly, is a close approximation to the number of machine instructions

- Initialize Display
- Start Display
- Check Status of Display
- Stop Display
- Insert x or Δx
- Insert y or Δy
- Insert sign
- Insert intensity bit
- Insert jump address
- Pentracking
- Rotation
- Translation
- Line blink
- Change Intensity
- Push jump
- Pop jump
- 10 bit addition
- 10 bit sbutraction
- 10 bit multiply
- 10 bit divide
- Address addition
- Address subtraction
- Test bits
- Set bits
- Shift (6 places)
- Save computer status
- Dispatch interrupt
- Restore computer status
- Set up teletype I/O
- Set up dataphone I/O
- Set up disk or drum I/O
- Set up paper tape I/O
- I/O Conversions
  - ASCII to Integer
  - ASCII to Display Code
  - Integer to ASCII
- Iterate
- Move a word
- Obtain current X and Y position
- Floating point (30 to 40 bits)
  - Add
  - Subtract
  - Multiply
  - Divide

Table 5-1

Display Instruction List

required by each display instruction. Whether the instructions are completely correct is actually irrelevant.

This method also requires that the weights $\{w_i\}$ be determined, which necessitates an excellent knowledge of the proposed display application and the manner in which a user will interact with the display console. As was first emphasized in Section 2. 3, the better these weights are estimated, the better will be the results produced by the display system model. Put another way, "You can't get something for nothing."

One of the nicest features of this method is the ease with which hardware features added to a display control-remote computer are accomodated. For example, if hardware light pen tracking is implemented, τ_i for tracking becomes zero, because no processing time is then taken to perform that function, and the instruction execution rate for the equipment increases accordingly. And of course none of the other τ's are affected. It is through this method that hardware-software tradeoffs are quantitatively treated.

5. 4 Final Analysis

The purpose of this section will be to unify the evaluation techniques discussed in Sections 5. 1 and 5. 3. Using these techniques it is possible to evaluate any display control remote computer to determine its effectiveness with respect to a specific application. In addition, it will be possible to include in the evaluation configurations with multiple consoles and display controls; specifically R consoles and N display controls, with $N \leq R$. The display controls will be constrained to be identical, which is certainly desirable for reasons of programming, interfacing, and maintenance. Also, only certain display control-remote computer configurations, called "balanced display configurations", will be acceptable. A balanced display configuration occurs when each display control drives an equal number of display consoles. Other configurations are termed "unbalanced". Figures 5-6 and 5-7 illustrate both cases for $R = 4$. Unbalanced configurations are not considered because they would normally not occur in practice for non-experimental systems.

QR, the quantity of information displayable on R consoles has been defined by eqn. 5. 1. 1. Now for each display control-remote computer being considered, and for each balanced configuration, a characteristic 3-tuple $(C_j, QRN_j, X4_j)$ is formed. C_j is the equipment's cost in dollars per month excluding core memory, QRN_j is the amount of information displayed on R consoles using N display controls, and $X4_j$ is the instruction execution rate from equation 5. 2. It will be assumed that $X4_j$ is the same for a particular remote computer-display control, whether there be one, two, three, or four display controls used with the remote computer. This is not strictly true, because multiple display controls will tend to simplify some of the processing which must be done by

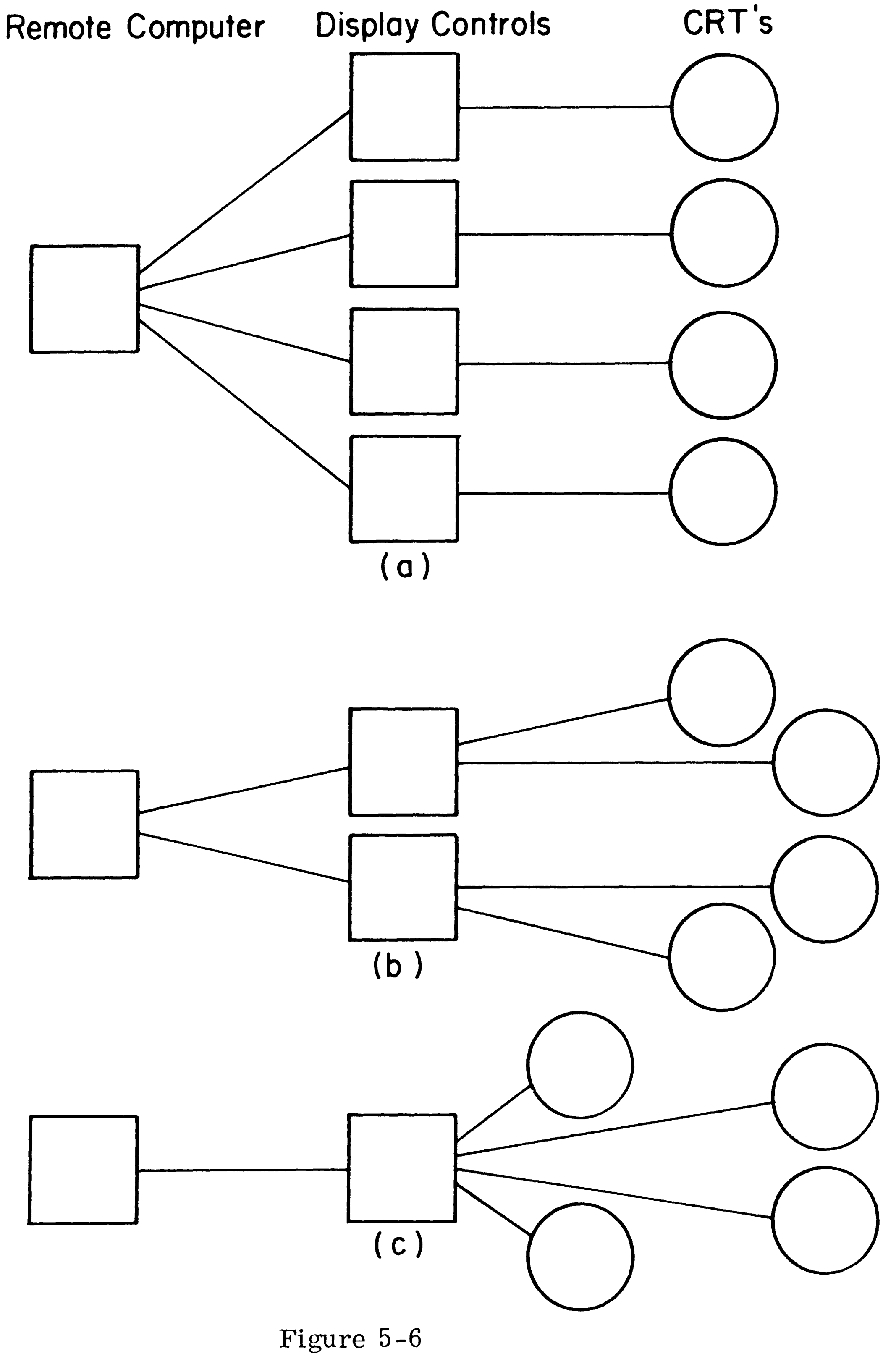


Figure 5-6

Balanced Display Configurations

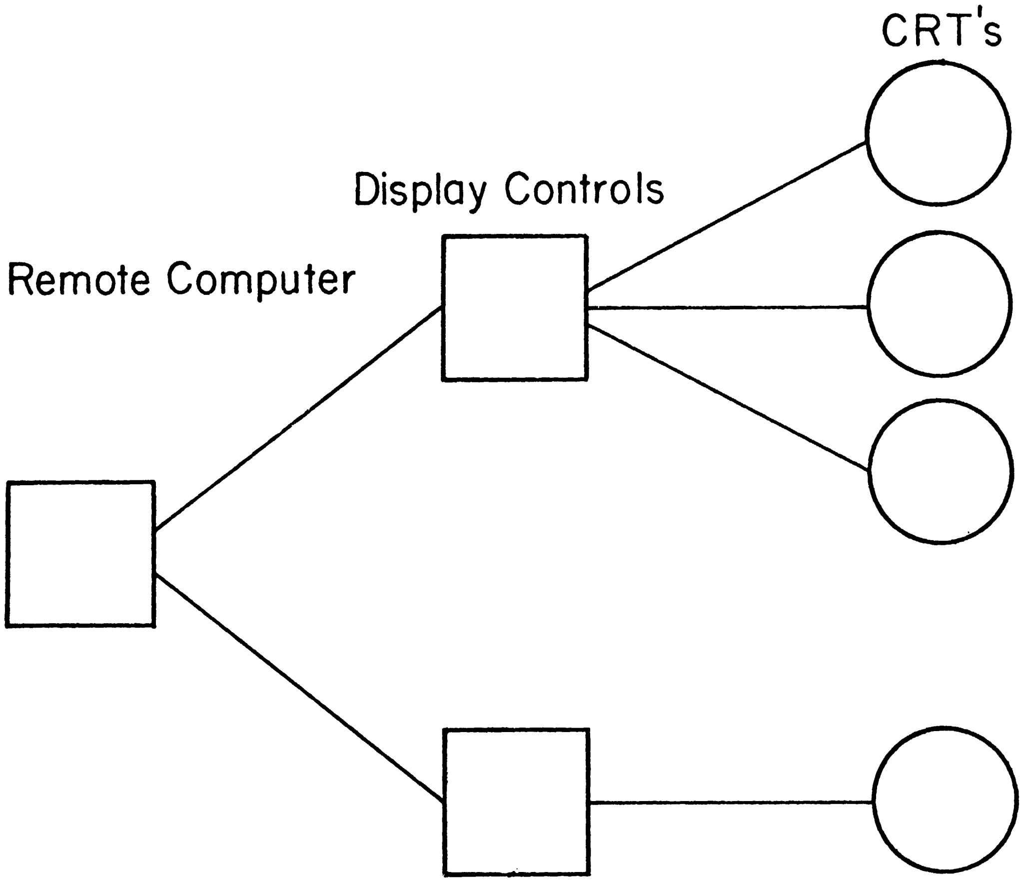


Figure 5-7

Unbalanced Display Configuration

the remote computer. Also, the effect on $X4_j$ of core memory cycle stealing by the display control has not been considered. The third quantity, QRN_j, is found from equation 5. 1 with R replaced by R/N, because each display control drives only R/N display consoles. That is, $QRN_j = \frac{Q}{x(\frac{R}{N} - 1) + 1}$ for configuration j.

Certain of these 3-tuples can be eliminated from further consideration on the basis of cost-effectiveness and also for not meeting requirements. Let Q_{min} be the minimum acceptable information displayed per console for a specific application, based again on the method of Section 5. 1. Then for all display control-remote computer subsystems for which $QRN_j < Q_{min}$, 3-tuple j must be discarded. Also, if $C_j > C_i$ and $X4_j \leq X4_i$, 3-tuple j is discarded because 3-tuple i describes a subsystem which costs less, yet has a higher instruction execution rate than subsystem j. If in addition it should happen that $C_j = C_i$, $X4_j = X4_i$, and $QRN_j < QRN_i$, subsystem j can be eliminated because subsystem i provides more display capacity for the same price. This algorithm is illustrated in Figure 5-8.

Those subsystems remaining will be such that $C_i > C_j \Leftrightarrow X4_i > X4_j$. This means that an increase in cost implies an increase in instruction execution rate. An example of such subsystems is given by Figure 5-9, for which x= 1, R= 3, $Q_{min} = 60\%$ with all five of the test patterns

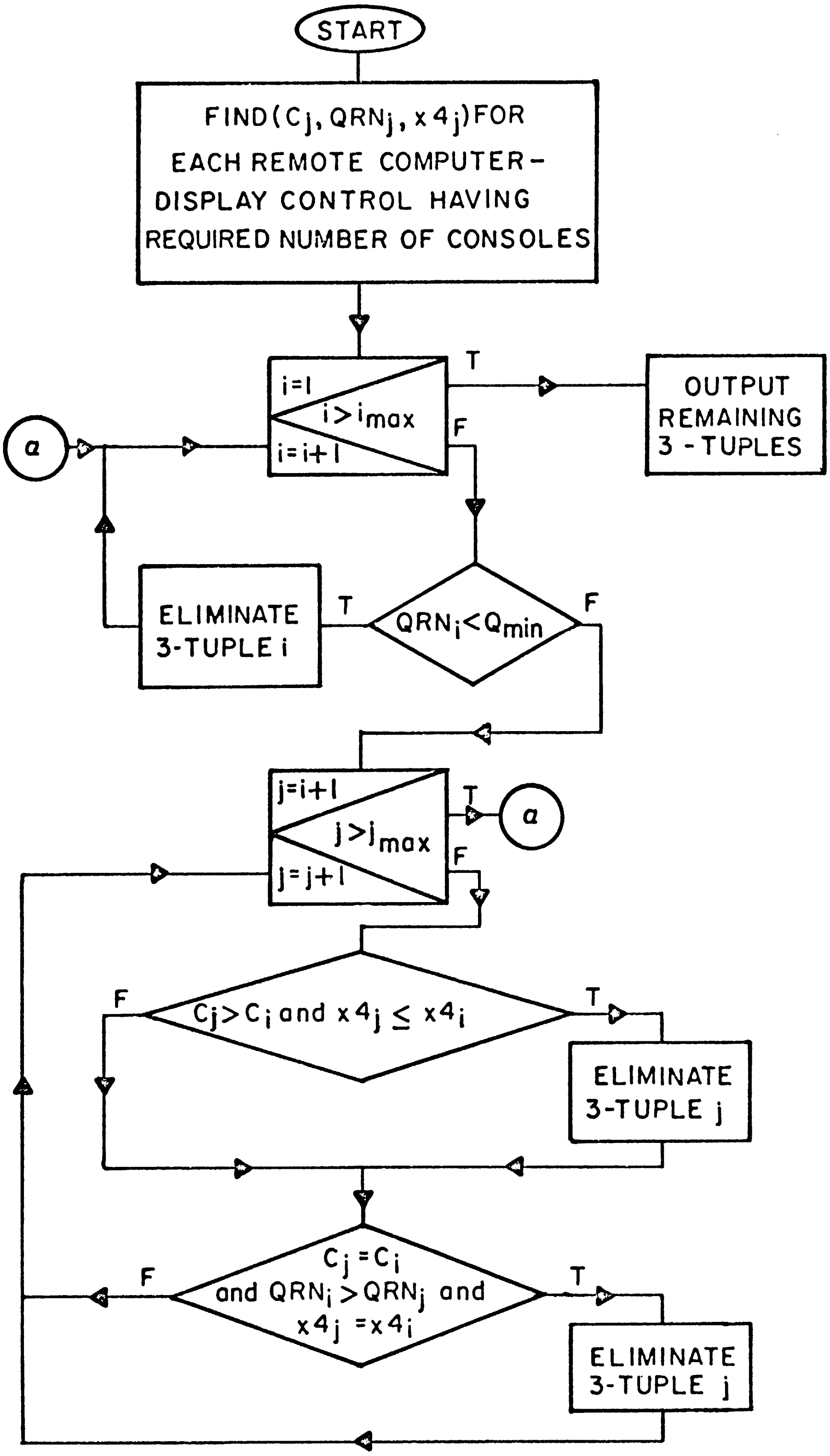


Figure 5-8

Selection of Remote Computer-Display Controls

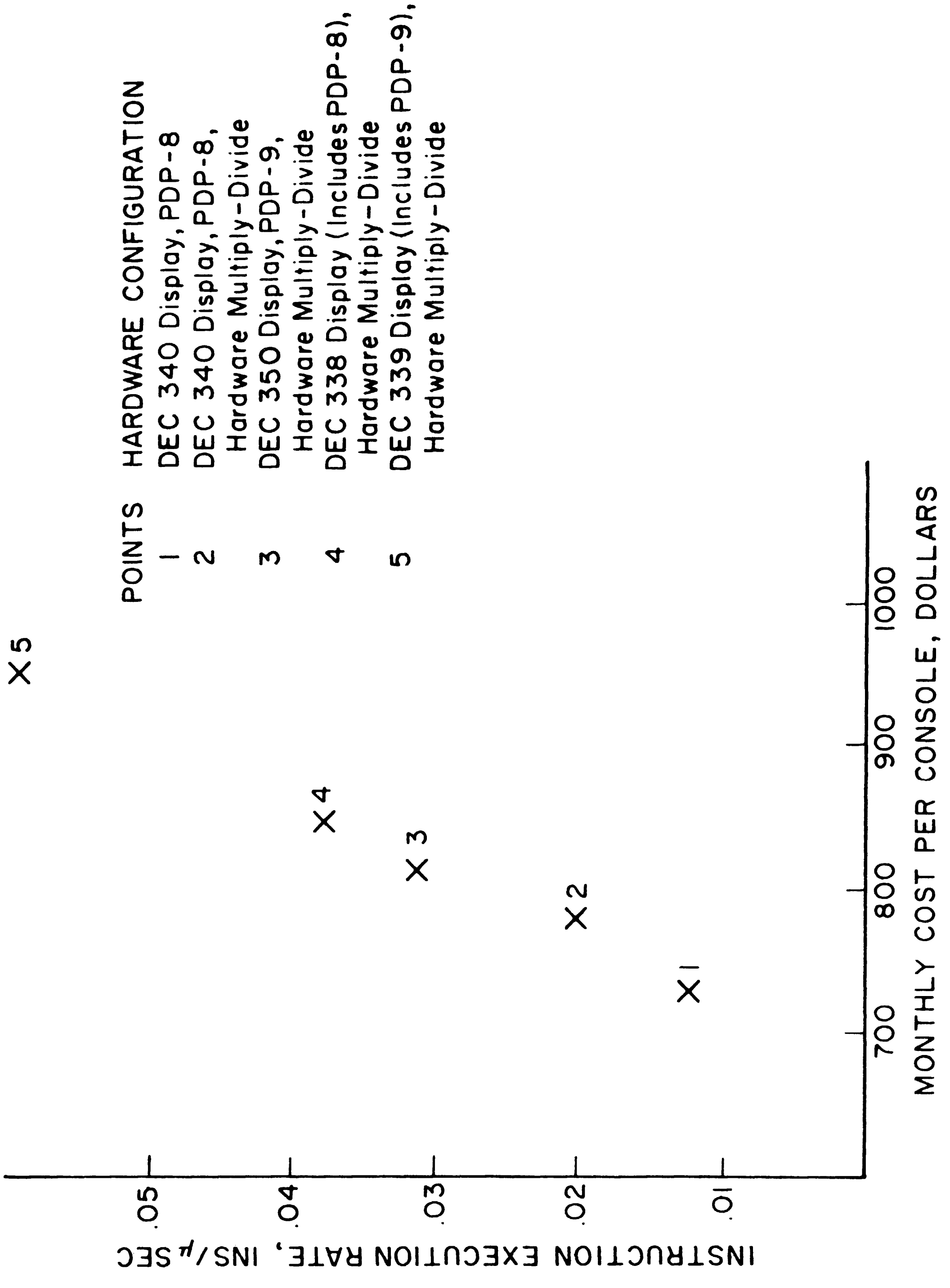


Figure 5-9 Example of Selected Remote Computer-Display Controls

equally weighted, and all w_i equal. The specific hardware for each configuration is tabulated. Only DEC display units and computers were considered in this example.

One of the subsystems found in this manner will be subsequently selected for inclusion in an optimal display system by the optimization algorithm presented in Chapter 4.

A computer program, called PREPROCESSOR, has been written to implement the above procedure. This program accepts as inputs a description of many remote computer-display controls, and a description of a display system application. These descriptions are in terms of the parameters discussed in the preceding sections; Q_i, $i = 1, 2, \ldots, 5$; Ω_i, $i = 1, 2, \ldots, 5$; R; N; x; τ_i, $i = 1, 2, \ldots, 42$; and w_i, $i = 1, 2, \ldots, 42$. The output is in a form suitable for input to the optimization programs. A complete discussion of PREPROCESSOR is available in Appendix A.

Chapter VI

APPLICATION

In the preceding chapters several tools were developed. In Chapters II and III, a display system model was presented. The model can be used to predict the response time of a particular display system when certain characteristics of both the system's hardware and application are known or can be estimated. An optimization algorithm was developed in Chapter IV. This algorithm uses the display system model, along with a description of a display system's proposed application and a selection of hardware subsystems, to configure a display system which minimizes response time subject to a cost constraint. Finally, Chapter V describes a means of evaluating the large number of available remote computers and display controls so that only those which are adequate for a given application will be considered in the optimization.

In this chapter these previously developed tools will be used to select optimum display hardware configurations for several diverse applications. The results of the optimizations will be useful in several ways. First, the wide range in system response time of optimal versus nonoptimal display systems will be illustrated. One of the motivations for the work reported here is that since this range exists, display systems' designs should be on the fast response side of the range. Second, the use of the tools in providing cost-effectiveness information for use in selecting which or several optimum (for different costs) display systems to install will be shown. Third, it will be shown that using multiple display consoles per remote computer can result in lower per console cost with equal response times. Fourth, an attempt will be made to deduce from the results some general guidelines or statements which will be useful in and of themselves to designers of graphical display systems. One point which will be dwelt on is determining under what circumstances the use of special-purpose hardware for image rotation is justified. Finally, the division of processing between the main and remote computers will be discussed.

6.1 Display System Hardware

In this chapter, display systems will be designed by selecting four subsystems (data link, core storage, bulk storage, remote computer-display control) from among many alternatives. A set of four subsystems is represented by the vector $X=(X1, X2, X3, X4)$. This section will describe the various possible subsystems and discuss their capabilities and costs. Mention of certain manufacturers' products in this and later sections is not to be considered an endorsement. The author has simply used as examples products with which he is familiar.

6.1.1 Data Link

A broad spectrum of data transmission facilities are available from The Michigan Bell Telephone Company. Similar services are also available for interstate service over the nation-wide telephone system. The pertinent information concerning these services can be found in Table 6-1. Also included in the table is the total cost of a ten mile data transmission link. This link includes two modems plus the actual transmission line required. As can be seen, the lowest and highest transmission rates are separated by three orders of magnitude in speed, and by two orders of magnitude in cost. The optimization program uses entries in the "maximum bit rate/sec" column as X1, the data link transmission rate.

6.1.2 Remote Computer Core Storage

Most small computers, such as those which will be described in Section 6.1.4, are capable of using up to 32,768 words of core storage. The prices of additional core storage for Digital Equipment Corporation's PDP-8 computer have been taken as typical. On a monthly basis, the prices are approximately $250 for the first additional core modules (4096 12-bit words), and $200 for the remaining modules. This information is given in Table 6-2. Notice that at least the first four blocks of storage (equivalent to a core module) are always included in the display system; otherwise the remote computer would have no core storage at all, and could not

Modem Type	Maximum Bit Rate/Second	Approximate Modem Cost Per Month	Transmission Line Cost Per Month	Total Monthly Cost of Ten Mile Link
103	300	$ 25	$8. 75 per station plus tolls	$ 67. 50
202	1200	$ 40	$8. 75 per station plus tolls	$ 97. 50
201A	2000	$ 70	$8. 75 per station plus tolls	$ 157. 50
201B	2400	$ 70	$3. 00 first 1/4 mile* $1. 00 additional 1/4 miles	$ 182. 00
Telpak A	40, 800	$ 250	$15/mile*	$ 650. 00
Telpak B	75, 000	$ 400	$20/mile*	$1000. 00
Telpak C	125, 000	$ 550	$25/mile*	$1350. 00
Telpak D	500, 000	$1300	$45/mile*	$3050. 00

Table 6-1

Michigan Intrastate Data Transmission Services

* leased line

monthly cost	storage capacity, blocks	description
200	4	1 core module
450	8	2 core module
650	12	3 core module
850	16	4 core module
1050	20	5 core module
1250	24	6 core module
1450	28	7 core module
1650	32	8 core module

Table 6-2

Remote Computer Core Storage

operate. At least these first four blocks are an integral part of any small computer. It is the next four blocks which constitute the first additional core module. Accordingly, the incremental cost of going from four to eight blocks is $250. The optimization program assigns values from the "storage capacity, blocks" column to the variable X2.

6. 1. 3 Remote Computer Bulk Storage

Only a subset of possible bulk storage devices are considered for use with the remote computer. These are the fixed head-per-track rotating storage media of drums and disks. Their fast access time and low cost (for small units) make them ideal for storing at the remote computer frequently used program and data files. Tape storage units are much slower (and often more expensive), while their larger capacity is usually not needed. Inexpensive small movable head disk units are not available, as such devices are economically feasible only for storing quite large amounts of information.

A quick survey of available disk and drum storage devices reveals that those marketed by Digital Equipment Corporation are among the less expensive. Table 6-3 gives information on four inexpensive disk memories with a wide range of capabilties. Entries in the column "storage capacity, blocks" are assigned to the variable X3 by the optimization program. Because the remote computer

Monthly Cost	Storage Capacity, Blocks	Description
0	0	No bulk storage
150	32	DEC DF32
225	64	DEC DF32 plus 1 expander
300	96	DEC DF32 plus 2 expanders
350	512	DEC RS08

Table 6-3

Remote Computer Bulk Storage

might not have any bulk storage, the first entry in Table 6-3 allows for this possibility.

6.1.4 Remote Computer-Display Control

There are currently available several dozen small computers which might be used as a remote computer in a display system. There are also available at least a dozen display controls, each of which could conceivably be used with any remote computer. Furthermore, the computers and display controls more often than not have many optional features, such as hardware multiply-divide. Taking all equipment combinations together amounts to hundreds or even thousands of possibilities. However, many small computers are fortunately functionally (and economically) very similar. The same is true of display controls. Accordingly, subsets of each group have been chosen for use here.

Individual computers considered are the PDP-8 [17] and PDP-9 [15]. The display controls studied are the DEC340 [16], IDI 10000 [25], and IDI11000 [25]. In addition, the DEC 338 [18], DEC 339 [18], Information Displays, Inc. IDIIOM [26], and Adage's AGT-10 [1], AGT -30 [1], and AGT-50 [1] combined remote computer-display controls are included. Optional features chosen for study are hardware multiplication and division in the remote computer (often called extended arithmatic element, or EAE), and analog three dimensional rotation-translation hardware in the display control (often called a matrix multiplier). These options are actually standard equipment for the Adage equipment. The rotation-translation equipment is not offered for any but the Adage systems. It was assumed that it could be made available by the other manufacturers for $10, 000.

The combinations of this equipment result in 39 different remote computer-display controls, each having unique characteristics. Table 6-4 lists each of the combinations. However, this is still not the end of the matter. Each remote computer-display control can drive several display consoles. Indeed, one remote computer might have attached several display controls, each in turn driving one or more display consoles. This matter was first discussed in Section 5. 4. Allowing no more than four display consoles, the eight possibilities tabulated in Table 6-5 arise. Combining these eight combinations with the 39 of Table 6-4 should yield a grand total of 8 x 39 = 312 combinations. This is not quite the case, as certain of the remote computer-display controls are not available in all eight configurations. After accounting for this, a total of 284 combinations remain. It is the function of the program PREPROCESSOR, discussed initially in Section 5. 4 and in greater detail in Appendix A, to determine which of these 284 possibilities should be considered by the optimization program for inclusion in a display system for a specific application. A typical set of results

Remote Computer	Display Control	Hardware Multiply-Divide	Matrix Multiplier
	DEC 338	No	No
	DEC 338	No	Yes
	DEC 338	Yes	No
	DEC 338	Yes	Yes
	DEC 339	No	No
	DEC 339	No	Yes
	DEC 339	Yes	No
	DEC 339	Yes	Yes
PDP-8	DEC 340	No	No
PDP-8	DEC 340	No	Yes
PDP-8	DEC 340	Yes	No
PDP-8	DEC 340	Yes	Yes
PDP-9	DEC 340	No	No
PDP-9	DEC 340	No	Yes
PDP-9	DEC 340	Yes	No
PDP-9	DEC 340	Yes	Yes
PDP-8	IDI10000	No	No
PDP-8	IDI10000	No	Yes
PDP-8	IDI10000	Yes	No
PDP-8	IDI10000	Yes	Yes
PDP-8	IDI11000	No	No
PDP-8	IDI11000	No	Yes
PDP-8	IDI11000	Yes	No
PDP-8	IDI11000	Yes	Yes
PDP-9	IDI10000	No	No
PDP-9	IDI10000	No	Yes
PDP-9	IDI10000	Yes	No
PDP-9	IDI10000	Yes	Yes
PDP-9	IDI11000	No	No
PDP-9	IDI11000	No	Yes
PDP-9	IDI11000	Yes	No
PDP-9	IDI11000	Yes	Yes
	IDIIOM	No	No
	IDIIOM	No	Yes
	IDIIOM	Yes	No
	IDIIOM	Yes	Yes
	AGT-10	Yes	Yes
	AGT-30	Yes	Yes
	AGT-50	Yes	Yes

Table 6-4

Remote Computer-Display Control Configurations

Number of Display Controls	Number of Display Consoles
1	1
1	2
1	3
1	4
2	2
2	4
3	3
4	4

Table 6-5

Possible Combinations of Display Controls and Display Consoles from PREPROCESSOR are given in Table 6-6. The optimization program uses the reciprocal of quantities in the "Average instruction execution time, μsec" as the variable X4, which is then the average instruction execution rate for the remote computer-display control.

The remote computers and display controls studied here cover a range of capabilities. A brief description of the equipment follows, so that the reader can be familiar with a few specifics.

The PDP-8 is a 12-bit computer with a 1.5 μsec core cycle time. While 32,768 words of core can be used, only 4096 are directly addressable. This can create time-consuming problems with programs requiring more than 4096 words of instructions and data. Because of the small word size, there are only six memory access instructions. There are no index registers, although certain core locations are automatically incremented after being used as an indirect address. There are many instructions to manipulate the accumulator.

The PDP-9, on the other hand, is an 18 bit machine with 1.0 μsec core cycle time. The long word length allows convenient referencing of up to 32,768 core locations, and also permits a goodly number of memory access instructions. Indexing is similar to the PDP-8.

Cost Per Month	Average Instruction Execution Time, μsec	Description
1175	272	PDP-8 and DEC 340
1862	271	PDP-8 and DEC 340 and EAE
1925	164	PDP-9 and DEC 340
2107	137	IDIIOM
2170	136	IDIIOM and EAE
3407	112	Adage AGT 50

Table 6-6

Typical Remote Computer-Display Controls Selected For Use in Optimization

The DEC 338 display terminal uses a PDP-8 and a display control with an extensive set of instructions allowing the control to operate independently of the PDP-8 between user interactions. Its capabilities include display subroutining, conditional transfers, and several display modes. Vector generation, however, is by slow digital means, so that the quantity of displayed information is limited. The DEC 339 system uses a PDP-9 in place of the PDP-8, but the display control is essentially unchanged, and it uses only 12 bits from each 18 bit PDP-9 word.

The Adage AGT-10, AGT-30, and AGT-50 display terminals all use the same 30 bit, 2 μsec computer. Both hardware multiply-divide and graphical translate-rotate are standard equipment, as is an indexing facility. On the AGT-10 and AGT-30, the display control does nothing but draw individual points or lines. The computer must continually load several output registers to keep the display operating. This takes at least fifty percent of the computer's time whenever a display is being generated. The AGT-50's display control, on the other hand, can display any sequence of lines or points which are stored contiguously in the computer's core memory. Fast analog vector generation is used in all three models.

The IDIIOM display terminal uses a 16 bit, 1.8 μsec computer with two accumulators and an index register. The display control's instruction set is nearly as powerful as the DEC 338's, and a fast analog vector generation system is used.

Of the three separate display controls considered, the DEC 340 uses digital vector generation, and the IDI 10000 and IDI 11000 use analog vector generation. Of the two IDI display controls, the 10000 is the faster. Both IDI controls were assumed to be versions essentially program compatible with the DEC 338. The DEC 340 is slightly less powerful than the DEC 338, and it can draw flicker free somewhat less information than the 338. Detailed statistics on all of these systems can be had in Appendix A.

6.2 Applications

Four display system applications have been selected for close examination. They are text editing, general two-dimensional drawing, general three-dimensional drawing, and general network analysis. These applications were chosen for their differences. That is, they each use the facilities of a display system in different ways. Short descriptions of each follow, to point out their characteristics.

6.2.1 Text Editing

In this application the display console and an associated typewriter keyboard is used first to enter text into the computer system, and then to scroll through the text to make corrections, additions, and deletions. The light pen is used to indicate what words or letters are to be modified, much as a cursor is used in some systems. As this is a very simple application, none of the more sophisticated remote computer-display controls' extra capabilities are used, and many of the display instruction mix's instructions are not used, as is seen in the appropriate column of Table 6-11. This Table gives an estimate of the probability of using each of the 42 display instructions. These probabilities are used to calculate the average display instruction execution time for these remote computer-display controls being considered for use in the optimization, as was discussed in Section 5.4.

Because only text is displayed on the display console, the weights Ω_i (Section 5.1) listed in Table 6-12 have been picked accordingly. Also in this Table is Q_{min} (Section 5.4) for each application. These weights and Q_{min} are used to determine a particular display control's suitability for a given application. Table 6-13 lists other pertinent characteristics of the text editing application. These are input parameters to the optimization for use with the display system model. Finally, Table 6-7 lists the various interactions which a user of the display system would be expected to create. The accompanying tabulation of the quantities N_i, π_i, B_i^M, and B_i^R (Section 2.3) are used to determine parameters of the display system model. The three rightmost columns are results which will be discussed in Section 6.6.

N_i	π_i	B_i^M	B_i^R	DESCRIPTIONS	One User	Two Users	Three Users
1000	.001	4	6	CALL UP A FILE & DISPLAY	R	R	R
100	.010	3	4	SCROLL	R	R	R
300	.359	2	3	ADD LINES TO DISPLAY & FILE	R	R	R
1000	.030	0	0	USE L. P. TO INDICATE LINE AND TEXT TO EDIT	R	R	R
300	.600	2	3	ENTER REPLACEMENT TEXT	R	R	R

Table 6-7

Text Editing Interactions

It is important to note that none of the quantities listed in these tables has been measured; they have been estimated. The point has been made before, and it is made again here: the better the estimates, the better the optimization results will be. This is true for all parameters and all applications, the second of which follows.

6.2.2 Two-Dimensional Drawing

This application is intended to provide a general two-dimensional drawing capability for a multitude of users. The user can either recall an old drawing from a library and do modifications, or start with a blank screen, build up a picture, name it, and store it in the library. Pictures consist of elements, which can be points, lines, text, or arcs of circles. These elements can be individually positioned into place, temporarily deleted, or permanently deleted. In addition, the entire picture can be translated, and its size can be arbitrarily scaled up or down. The light pen, naturally, is used for indicating elements and positions. A keyboard is used for entering text elements and picture names. Statistical information concerning this application is available in Tables 6-8, 6-11, 6-12, and 6-13.

6.2.3 Three-Dimensional Drawing

Aside from the addition of a third dimension, the only substantial external difference between this and the previous application

N_i	π_i	B_i^M	B_i^R	DESCRIPTION	One User	Two Users	Three Users
10	.001	0	1	TERMINATE EXECUTION	R	R	R
50	.005	0	1	DELETE CURRENT PICTURE	R	R	R
100	.005	1	2	GET MENU OF EXISTING PICTURES	M	M	M
100	.005	2	3	PICK PICTURE FROM MENU	M	M	M
100	.005	2	3	CREATE NEW PICTURE	M	M	M
200	.005	1	2	NAME NEW PICTURE	M	M	M
100	.005	4	6	SAVE CURRENT PICTURE	M	M	M
50	.100	0	0	ENTER POINT	R	R	R
60	.369	0	0	DRAW LINE FROM PREVIOUS POINT	R	R	R
200	.050	0	1	ENTER TEXT ELEMENT	R	R	R
2000	.025	0	1	ENTER ARC OF CIRCLE	M	M	M
100	.075	0	0	DRAW NEW LINE	R	R	R

Table 6-8

2-D Drawing Interactions

N_i	π_i	B_i^M	B_i^R	DESCRIPTION	One User	Two Users	Three Users
100	.050	0	1	DELETE ELEMENT FROM PICTURE	R	R	R
110	.030	0	1	TEMPORARILY ERASE ELEMENT	R	R	R
500	.010	0	1	REDISPLAY ALL TEMP ERASED ELEMENTS	R	M	R
3000	.025	0	1	SCALE PICTURE	M	M	M
200	.025	0	1	TRANSLATE PICTURE	R	R	R
200	.070	0	11	MOVE ELEMENT	R	R	R
50	.150	0	0	IDENTIFY ELEMENT	R	R	R

Table 6-8 (continued)

2-D Drawing Interactions

N_i	π_i	B_i^M	B_i^R	DESCRIPTION	One User	Two Users	Three Users
10	.001	0	1	TERMINATE EXECUTION	R	R	R
50	.005	0	1	DELETE CURRENT PICTURE	R	R	R
100	.005	1	2	GET MENU OF EXISTING PICTURES	M	M	R
100	.005	2	3	PICK PICTURE FROM MENU	M	M	R
100	.005	2	3	CREATE NEW PICTURE	M	M	R
200	.005	1	2	NAME NEW PICTURE	M	M	R
100	.005	4	6	SAVE CURRENT PICTURE M	M	M	M
75	.100	0	0	ENTER POINT	R	R	R
85	.309	0	0	DRAW LINE FROM PREVIOUS POINT	R	R	R
225	.050	0	1	ENTER TEXT ELEMENT	M	R	R
3000	.025	0	1	ENTER ARC OF CIRCLE	M	M	R
125	.075	0	0	DRAW NEW LINE	R	R	R
100	.050	0	1	DELETE ELEMENT FROM PICTURE	R	R	R

Table 6-9

3-D Drawing Interactions

N_i	π_i	B_i^M	B_i^R	DESCRIPTION	One User	Two Users	Three Users
110	.030	0	1	TEMPORARILY ERASE ELEMENT	R	R	R
500	.010	0	1	REDISPLAY ALL TEMP ERASED ELEMENTS	M	R	R
3000	.025	0	1	SCALE PICTURE	M	M	R
225	.025	0	1	TRANSLATE PICTURE	M	R	R
4000	.050	0	1	ROTATE PICTURE	M	M	R
225	.070	0	1	MOVE ELEMENT	M	R	R
100	.150	0	0	IDENTIFY ELEMENT WITH L. P.	R	R	R

Table 6-9 (continued)

3-D Drawing Interactions

is the capability to rotate a solid object in three dimensions. Computationally, however, more effort is needed to keep track of the third dimension. Specifics can be found in Tables 6-9, 6-11, 6-12, and 6-13.

6.2.4 General Network Analysis

This is meant to represent any one of many network analysis programs. The network elements might be electrical components such as resistors, inductors, capacitors, transistors, and sources, or they might be mechanical devices such as springs, masses, and dashpots, or they might represent the stochastic service system elements of queues, servers, branches, and blocks. Whatever the case, the user progresses through several phases. First, a network consisting of elements and their connections is constructed. Elements are chosen from a menu and moved into place with the light pen, and connections drawn. Changes and deletions are made as required. In the next phase, numerical attributes are assigned each element. For electrical networks, this would correspond to ohms, henrys, farads, or a transistor type. In the third phase, the desired analysis results are specified. The network is then analyzed by whatever numerical techniques are appropriate to the type of network drawn, and finally, the requested results are plotted.

N_i	π_i	B_i^M	B_i^R	DESCRIPTION			
100	.038	1	2	PICK PHASE FROM MENU	M	M	M
75	.125	0	1	PICK ELEMENT FROM MENU	R	R	R
200	.125	0	1	MOVE ELEMENT INTO PLACE	R	R	R
50	.250	0	10	IDENTIFY AN ELEMENT WITH L. P.	M	M	M
200	.186	0	1	CONNECT PORTS OF IDENTIFIED ELEMENTS	R	R	R
100	.025	0	1	DELETE CONNECTION	R	R	R
100	.025	0	1	DELETE AN IDENTIFIED ELEMENT	R	R	R
200	.050	0	1	MOVE WHOLE NETWORK (TRANSLATE)	R	R	R
200	.125	0	1	ASSIGN VALUE TO IDENTIFIED ELEMENT	R	R	R
100	.012	1	2	PICK DESIRED OUTPUT FROM MENU	M	M	M
200	.012	0	1	INDICATE OUTPUT NODE(S) ON NETWORK	R	R	R
100000	.012	3	10	SOLVE PROBLEM	M	M	M
20000	.012	1	12	PREPARE OUTPUT	M	M	M

Table 6-10

Network Analysis Interactions

	Text Editing	2-D Drawing	3-D Drawing	Network Analysis
Initialize Display	.01	.01	.0063	.0067
Start Display	.01	.01	.0063	.0067
Check Status of Display	.01	.01	.0063	.0067
Stop Display	.01	.01	.0063	.0067
Insert x or Δ x	.01	.0025	.0015	.0016
Insert y or Δ y	.01	.0025	.0015	.0016
Insert Sign	.0	.0025	.0015	.0016
Insert Intensity Bit	.0	.0025	.0015	.0016
Insert Jump Address	.02	.0025	.0015	.0016
Pentracking	.1	.52	.3245	.3476
Rotation	.0	.0	.0625	.0
Translation	.0	.001	.0007	.0007
Line Blink	.0	.0	.0	.0
Change Intensity	.0	.0	.0	.0

Table 6-11

Display Instruction Mix

	Text Editing	2-D Drawing	3-D Drawing	Network Analysis
Push Jump	.0	.0	.0	.0
Pop Jump	.0	.0	.0	.0
10 Bit Addition	.05	.015	.0094	.01
10 Bit Subtraction	.05	.015	.0094	.01
10 Bit Multiply	.0	.005	.0031	.0033
10 Bit Divide	.0	.0	.0	.0
Address Addition	.025	.005	.0031	.0033
Address Subtraction	.025	.005	.0031	.0033
Test Bits	.025	.02	.0126	.0134
Set Bits	.025	.02	.0126	.0134
Shift (6 places)	.0	.02	.0126	.0134
Save Computer Status	.1	.065	.0413	.044
Dispatch Interrupt	.1	.065	.0413	.044
Restore Computer Status	.1	.065	.0413	.044

Table 6-11 (continued)

Display Instruction Mix

Set Up Teletype I/O	.1	.01	.0063	.0067
Set Up Dataphone I/O	.01	.01	.0063	.0067
Set Up Disk or Drum I/O	.01	.01	.0063	.0067
Set Up Paper Tape I/O	.0	.0	.0	.0
ASCII to Integer	.0	.0	.0	.0
ASCII to Display Code	.1	.005	.0031	.0033
Integer to ASCII	.0	.0	.0	.0
Iterate	.03	.765	.0478	.051
Move a Word	.03	.01	.0063	.0067
Obtain Current X and Y Position	.04	.01	.0063	.0067
Floating Add	.0	.0	.0781	.0833
Floating Subtract	.0	.0	.0781	.0833
Floating Multiply	.0	.0	.0781	.0833
Floating Divide	.0	.0	.0781	.0833

Table 6-11 (continued)

Display Instruction Mix

	Text Editing	2-D Drawing	3-D Drawing	Network Analysis
Alphanumeric Test Pattern (Figure 5-1)	1.0	0.0	0.0	0.1
Weather Map Test Pattern (Figure 5-2)	0.0	0.0	0.0	0.0
Graph Test Pattern Figure 5-3)	0.0	0.0	0.0	0.2
Architectural Drawing Test Pattern (Figure 5-4)	0.0	0.5	0.5	0.0
Electronic Schematic Test Pattern (Figure 5-5)	0.0	0.5	0.5	0.7
Q_{min}	100.0	75.0	150.0	60.0

Table 6-12

Display Weights Ω_i and Q_{min}

	Text Editing	2-D Drawing	3-D Drawing	Network Analysis
NPPPC	500	100.00	200.00	3000.00
MSGLTH	1280	500.00	800.00	4000.00
MAXPAG	200	200.00	200.00	200.00
SYSPAG	2	3.00	4.00	4.00
ARRIVE	2	0.20	0.20	0.10
UMC	1	0.25	0.25	0.25
PACESS(x)	$1-e^{\frac{-x}{20}}$	$1-e^{\frac{-x}{20}}$	$1-e^{\frac{-x}{20}}$	$1-e^{\frac{-x}{20}}$

Table 6-13

Application Characteristics

The application's statistics are contained in Tables 6-10, 6-11, 6-12, and 6-13. Note in Table 6-11 the heavy use of floating point operations, which is a consequence of the network analysis.

6. 3 Optimization Results

The optimization program was used for the display subsystems and applications presented in Sections 6. 1 and 6. 2, for one, two and three display consoles. Thus twelve cost versus minimum response time curves have been generated. It was felt that very little incremental knowledge could be gained from studying four console display systems. They were therefore not considered, even though the appropriate output information from PREPROCESSOR exists.

Tables 6-14, 6-15, 6-16, and Figure 6-1 give results for the text editing application. Tables 6-17, 6-18, 6-19, and Figure 6-2 report the two-dimensional drawing results, while Tables 6-20, 6-21, 6-22, and Figure 6-23 have the three-dimensional drawing results. Finally, the network analysis results are found in Tables 6-23, 6-24, 6-25, and Figure 6-5. In each case the tables give optimum display systems, with their average response time and monthly cost. In the tables following the description of each remote computer is a pair of digits. The first is the number of display controls; the second, the number of display consoles used with the computer. The figures graphically present per console cost versus minimum

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
1505	8.494	103 SERIES DATA LINK 1 CORE MODULE NO BULK STORAGE DEC 338 REMOTE DISPLAY TERM.	1, 1
1535	2.286	SWITCHED LINES - ASYNCHRONOUS 1 CORE MODULE NO BULK STORAGE DEC 338 REMOTE DISPLAY TERM.	1, 1
1595	1.458	SWITCHED LINES - SYNCHRONOUS 1 CORE MODULE NO BULK STORAGE DEC 338 REMOTE DISPLAY TERM.	1, 1
1963	0.073	SWITCHED LINES - ASYNCHRONOUS 1 CORE MODULE DEC RF08 + RS08 DISK UNIT DEC 339	1, 1
2980	0.035	103 SERIES DATA LINK 6 CORE MODULES DEC RF08 + RS08 DISK UNIT DEC 339	1, 1

Table 6-14

Text Editing, One User

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
3892	0.027	PRIVATE VOICE GRADE LINES 8 CORE MODULES DEC RF08 + RS08 DISK UNIT IDI INPUT OUTPUT MACHINE (IDIIOM)	1, 1

Table 6-14 (continued)

Text Editing, One User

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
2430	8.493	103 SERIES DATA LINK 1 CORE MODULE NO BULK STORAGE DEC 338 REMOTE DISPLAY TERM.	2,2
2460	2.286	SWITCHED LINES - ASYNCHRONOUS 1 CORE MODULE NO BULK STORAGE DEC 338 REMOTE DISPLAY TERM.	2,2
3421	0.058	PRIVATE VOICE GRADE LINES 3 CORE MODULES DEC RF08 + RS08 DISK UNIT DEC 339	2,2
4417	0.030	PRIVATE VOICE GRADE LINES 8 CORE MODULES DEC RF08 + RS08 DISK UNIT DEC 339	2,2

Table 6-15

Text Editing, Two Users

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
3604	8.497	103 SERIES DATA LINK 2 CORE MODULES NO BULK STORAGE DEC 338 REMOTE DISPLAY TERMINAL	3, 3
3634	2.289	SWITCHED LINES - ASYNCHRONOUS 2 CORE MODULES NO BULK STORAGE DEC 338 REMOTE DISPLAY TERMINAL	3, 3
3694	1.461	SWITCHED LINES - SYNCHRONOUS 2 CORE MODULES NO BULK STORAGE DEC 338 REMOTE DISPLAY TERMINAL	3, 3
3972	0.086	PRIVATE VOICE GRADE LINES 2 CORE MODULES DEC RF08 + RS08 DISK UNIT DEC 338 REMOTE DISPLAY TERMINAL	3, 3
4485	0.063	SWITCHED LINES - ASYNCHRONOUS 5 CORE MODULES DEC RF08 + RS08 DISK UNIT DEC 338 REMOTE DISPLAY TERMINAL	3, 3

Table 6-16

Text Editing, Three Users

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
5342	0.035	PRIVATE VOICE GRADE LINES 8 CORE MODULES DEC RF08 + RS08 DISK UNIT DEC 339	3, 3

Table 6-16 (continued)

Text Editing, Three Users

Figure 6-1

Minimum Response Times, Text Editing

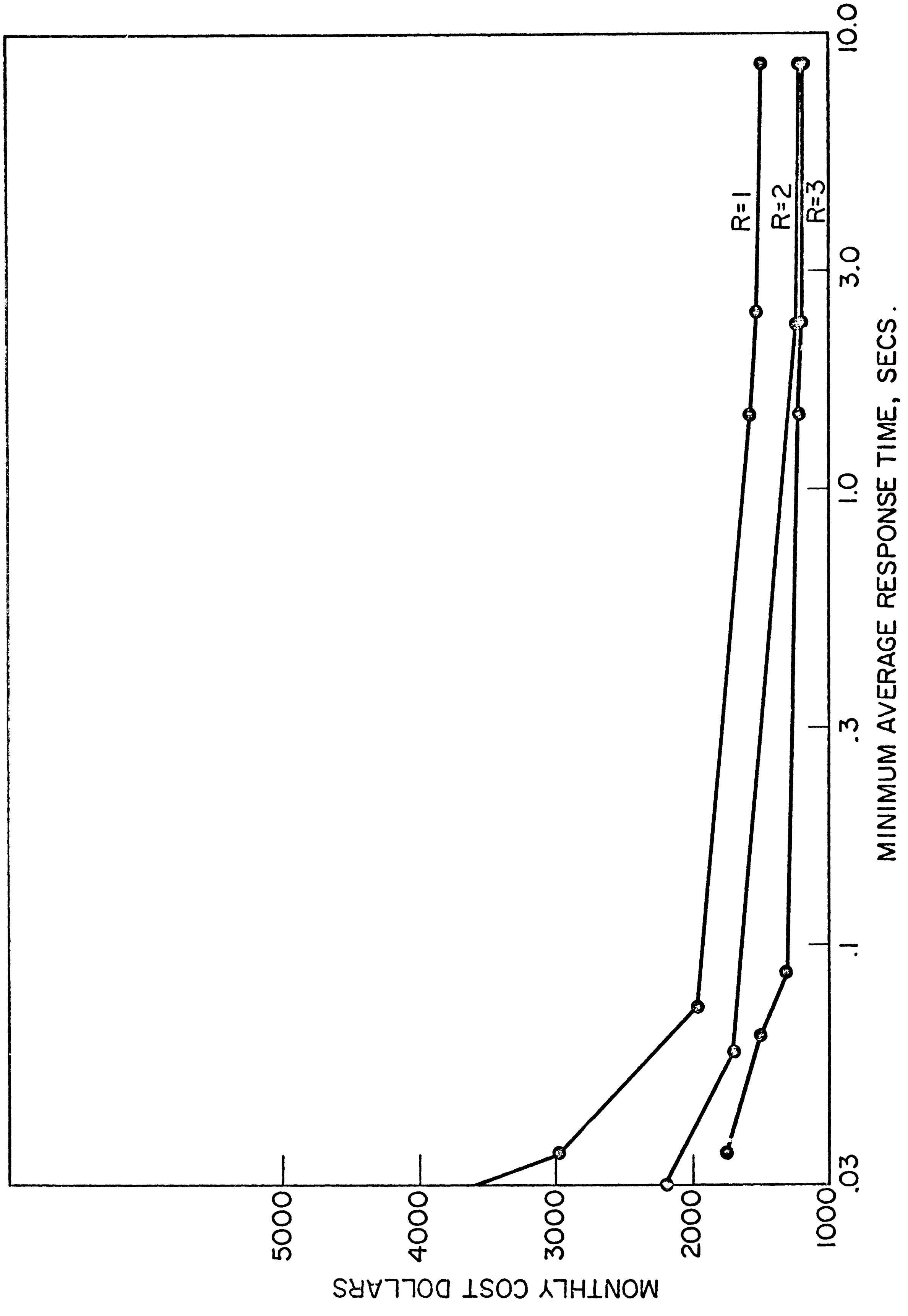

MONTHLY COST DOLLARS
5000
4000
3000
2000
1000
.03
.1
.3
1.0
3.0
10.0
MINIMUM AVERAGE RESPONSE TIME, SECS.
R=1
R=2
R=3

Figure 6-1

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
1330	0.281	103 SERIES DATA LINK 1 CORE MODULE NO BULK STORAGE PDP-8 + DEC 340 DISPLAY	1,1
1420	0.191	SWITCHED LINES - SYNCHRONOUS 1 CORE MODULE NO BULK STORAGE PDP-8 + DEC 340 DISPLAY	1,1
1474	0.076	103 SERIES DATA LINK 1 CORE MODULE ONE MODULE, DEC DF32 PDP-8 + DEC 340 DISPLAY	1,1
1922	0.025	PRIVATE VOICE GRADE LINES 1 CORE MODULE DEC RF08 + RS08 DISK UNIT ADAGE AGT 10	1,1
2390	0.017	TELPAK A 1 CORE MODULE DEC RF08 + RS08 DISK UNIT ADAGE AGT 10	1,1

Table 6-17

Two-Dimensional Drawing, One User

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
2998	0. 013	TELPAK A 4 CORE MODULES DEC DF32, 1 DS32 EXPANDER ADAGE AGT 10	1, 1
3986	0. 012	TELPAK B 7 CORE MODULES DEC RF08 + RS08 DISK UNIT ADAGE AGT 10	1, 1

Table 6-17 (continued)

Two-Dimensional Drawing, One User

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
2329	0.282	103 SERIES DATA LINK 2 CORE MODULES NO BULK STORAGE PDP-8 + DEC 340 DISPLAY	2, 2
2359	0.102	SWITCHED LINES - ASYNCHRONOUS 2 CORE MODULES NO BULK STORAGE PDP-8 + DEC 340 DISPLAY	2, 2
2473	0.076	103 SERIES DATA LINK 2 CORE MODULES ONE MODULE, DEC DF32 PDP-8 + DEC 340 DISPLAY	2, 2
2967	0.029	SWITCHED LINES - ASYNCHRONOUS 2 CORE MODULES DEC DF32, 1 DS32 EXPANDER IDIIOM + EXTENDED ARITHMETIC	1, 2
3474	0.018	TELPAK A 3 CORE MODULES DEC DF32, 1 DS32 EXPANDER PDP-9 + DEC 340 DISPLAY	2, 2

Table 6-18

Two-Dimensional Drawing, Two Users

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
3959	0. 015	TELPAK A 4 CORE MODULES DEC RF08 + RS08 DISK UNIT IDIIOM + EXTENDED ARITHMETIC	1, 2
4997	0. 013	TELPAK A 3 CORE MODULES DEC RF08 + RS08 DISK UNIT ADAGE AGT 50	1, 2
5945	0. 012	TELPAK A 5 CORE MODULES DEC RF08 + RS08 DISK UNIT ADAGE AGT 50	1, 2

Table 6-18 (continued)

Two-Dimensional Drawing, Two Users

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
* 3079	0. 294	103 SERIES DATA LINK 2 CORE MODULES NO BULK STORAGE PDP-8 + DEC 340 DISPLAY	3, 3
3194	0. 173	PRIVATE VOICE GRADE LINES 2 CORE MODULES NO BULK STORAGE PDP-8 + DEC 340 DISPLAY	3, 3
3493	0. 048	PRIVATE VOICE GRADE LINES 2 CORE MODULES DEC DF32, 1 DS32 EXPANDER PDP-8 + 340 DISPLAY + EXTENDED ARITHMETIC	3, 3
3992	0. 029	103 SERIES DATA LINK 2 CORE MODULES DEC DF32, 1 DS32 EXPANDER DEC 339	3, 3

Table 6-19

Two-Dimensional Drawing, Three Users

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
4474	0. 021	TELPAK A 3 CORE MODULES DEC DF32, 1 DS32 EXPANDER PDP-9 + DEC 340 DISPLAY	3, 3
4973	0. 015	TELPAK A 4 CORE MODULES DEC DF32, 1 DS32, EXPANDER DEC 339	3, 3
5962	0. 014	TELPAK B 7 CORE MODULES DEC RF08 + RS08 DISK UNIT DEC 339	3, 3

Table 6-19 (continued)

Two-Dimensional Drawing, Three Users

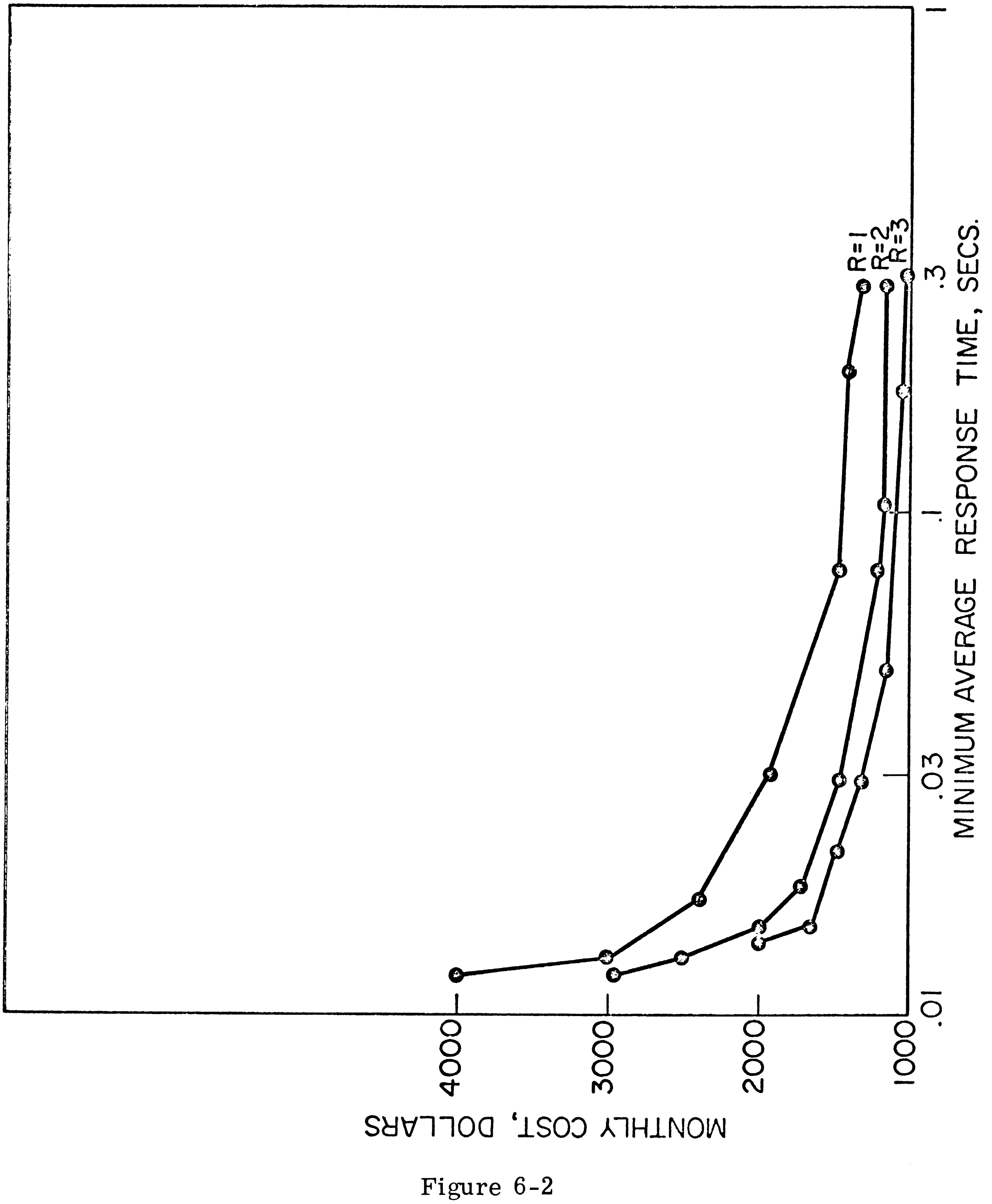


Figure 6-2

Minimum Response Times, Two-Dimensional Drawing

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
2082	1.553	103 SERIES DATA LINK 1 CORE MODULE NO BULK STORAGE IDI INPUT OUTPUT MACHINE (IDIIOM)	1, 1
2112	0.433	SWITCHED LINES - ASYNCHRONOUS 1 CORE MODULE NO BULK STORAGE IDI INPUT OUTPUT MACHINE (IDIIOM)	1, 1
* 2196	0.246	PRIVATE VOICE GRADE LINES 1 CORE MODULE NO BULK STORAGE IDI INPUT OUTPUT MACHINE (IDIIOM)	1, 1
2471	0.072	PRIVATE VOICE GRADE LINES 1 CORE MODULE DEC DF32, 1 DS32 EXPANDER IDIIOM + EXTENDED ARITHME ARITHMETIC	1, 1

Table 6-20

Three-Dimensional Drawing, One User

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
2977	0. 029	TELPAK A 1 CORE MODULE DEC RF08 + RS08 DISK UNIT IDIIOM + EXTENDED ARITHMETIC	1, 1
3962	0. 021	TELPAK C 1 CORE MODULE DEC DF32, 1 DS32 EXPANDER PDP-9 + IDI 10000 + EXTENDED ARITHMETIC	1, 1
4954	0. 017	TELPAK B 3 CORE MODULES ONE MODULE, DEC DF32 ADAGE AGT 50	1, 1

Table 6-20 (continued)

Three-Dimensional Drawing, One User

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
4101	0.739	103 SERIES DATA LINK 2 CORE MODULES NO BULK STORAGE DEC PDP-8 + IDI 10000	2, 2
* 4131	0.235	SWITCHED LINES - ASYNCHRONOUS 2 CORE MODULES NO BULK STORAGE DEC PDP-8 + IDI 10000	2, 2
4192	0.175	SWITCHED LINES - SYNCHRONOUS 2 CORE MODULES NO BULK STORAGE DEC PDP-8 + IDI 10000	2, 2
4985	0.038	TELPAK A 2 CORE MODULES DEC DF32, 1 DS32 EXPANDER PDP-8 + IDI 10000 + EXTENDED ARITH.	2, 2
5867	0.021	TELPAK C 2 CORE MODULES DEC RF08 +RS08 DISK UNIT PDP-9 + IDI 10000 + EXTENDED ARITH.	2, 2

Table 6-21

Three-Dimensional Drawing, Two Users

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
6964	0.020	TELPAK C 5 CORE MODULES DEC RF08+RS08 DISK UNIT PDP-9 + IDI 10000 + EAE + MATRIX	2,2

Table 6-21 (continued)

Three-Dimensional Drawing, Two Users

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
5738	1.631	103 SERIES DATA LINK 2 CORE MODULES NO BULK STORAGE DEC PDP-8 + IDI 10000	3, 3
5768	0.467	SWITCHED LINES - ASYNCHRONOUS 2 CORE MODULES NO BULK STORAGE DEC PDP-8 + IDI 10000	3, 3
* 5997	0.108	SWITCHED LINES - ASYNCHRONOUS 2 CORE MODULES ONE MODULE, DEC DF32 PDP-8 + IDI 10000 + EX. ARITH.	3, 3
6978	0.024	TELPAK A 3 CORE MODULES DEC RF08 + RS08 DISK UNIT PDP-9 + IDI 10000 + EX. ARITH.	3, 3
7967	0.021	TELPAK C 5 CORE MODULES ONE MODULE, DEC DF32 PDP-9 + IDI 10000 + EX. ARITH.	3, 3

Table 6-22

Three-Dimensional Drawing, Three Users

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
8826	0. 020	TELPAK C 5 CORE MODULES DEC RF08 + R508 DISK UNIT PDP-9 + IDI 10000 + EAE + MATRIX	3, 3

Table 6-22 (continued)

Three-Dimensional Drawing, Three Users

Figure 6-3

Minimum Response Times, Three-Dimensional Drawing

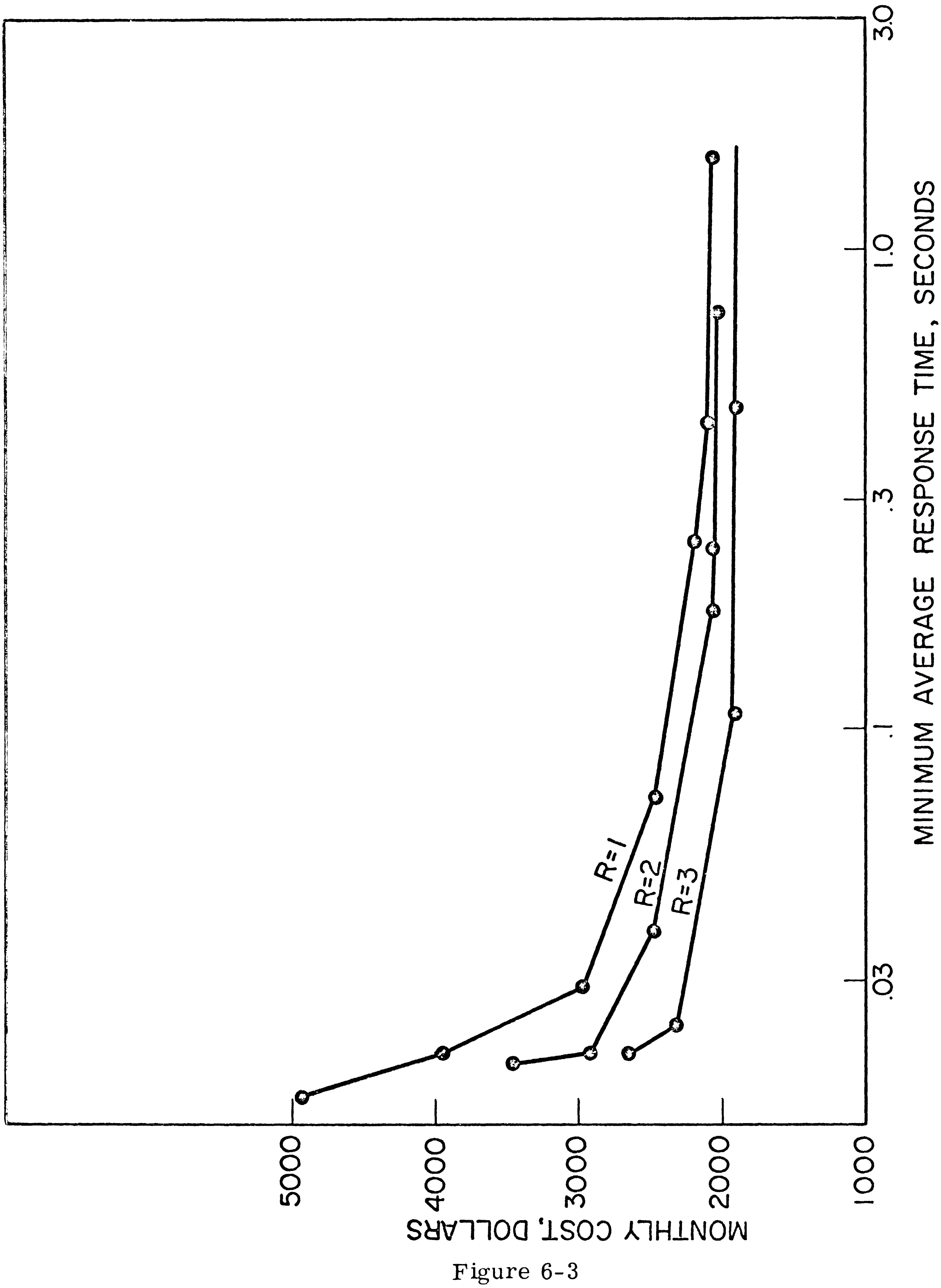

MONTHLY COST, DOLLARS
5000
4000
3000
2000
1000
MINIMUM AVERAGE RESPONSE TIME, SECONDS
.03
.1
.3
1.0
3.0
R=1
R=2
R=3

Figure 6-3

MONTHLY COST, DOLLAR	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
1557	8.906	103 SERIES DATA LINK 1 CORE MODULE NO BULK STORAGE ADAGE AGT 10	1,1
1588	2.426	SWITCHED LINES - ASYNCHRONOUS 1 CORE MODULE NO BULK STORAGE ADAGE AGT 10	1,1
1648	1.562	SWITCHED LINES - SYNCHRONOUS 1 CORE MODULE NO BULK STORAGE ADAGE AGT 10	1,1
1672	1.346	PRIVATE VOICE GRADE LINES 1 CORE MODULE NO BULK STORAGE ADAGE AGT 10	1,1
1925	0.201	PRIVATE VOICE GRADE LINES 1 CORE MODULE DEC RF08 + RS08 DISK UNIT ADAGE AGT 10	1,1

Table 6-23

Network Analysis, One User

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
2840	0.114	TELPAK A 3 CORE MODULES DEC RF08 + RS08 DISK UNIT ADAGE AGT 10	1, 1
3836	0.072	TELPAK A 8 CORE MODULES DEC RF08 + RS08 DISK UNIT ADAGE AGT 10	1, 1
4536	0.069	TELPAK C 8 CORE MODULES DEC RF08 + RS08 DISK UNIT ADAGE AGT 10	1, 1

Table 6-23 (continued)

Network Analysis, One User

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
2663	9.704	103 SERIES DATA LINK 2 CORE MODULES NO BULK STORAGE IDI INPUT OUTPUT MACHINE (IDIIOM)	1, 2
* 2694	2.884	SWITCHED LINES - ASYNCHRONOUS 2 CORE MODULES NO BULK STORAGE IDI INPUT OUTPUT MACHINE (IDIIOM)	1, 2
3491	0.181	PRIVATE VOICE GRADE LINES 4 CORE MODULES DEC RF08 + RS08 DISK UNIT IDIIOM + EXTENDED ARITHMETIC	1, 2
4359	0.100	TELPAK A 6 CORE MODULES DEC RF08 + RS08 DISK UNIT IDIIOM + EXTENDED ARITHMETIC	1, 2
5499	0.075	TELPAK B 8 CORE MODULES DEC RF08 + RS08 DISK UNIT PDP-9 + IDI 10000 + EX. ARITH.	1, 2

Table 6-24

Network Analysis, Two Users

	MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
	5738	10.218	103 SERIES DATA LINK 2 CORE MODULES NO BULK STORAGE DEC PDP-8 + IDI 10000	3, 3
	5769	3.157	SWITCHED LINES - ASYNCHRONOUS 2 CORE MODULES NO BULK STORAGE DEC PDP-8 + IDI 10000	3, 3
*	5853	1.978	PRIVATE VOICE GRADE LINES 2 CORE MODULES NO BULK STORAGE DEC PDP-8 + IDI 10000	3, 3
	5991	1.697	103 SERIES DATA LINK 2 CORE MODULES DEC RF08 + RS08 DISK UNIT DEC PDP-8 + IDI 10000	3, 3
	6978	0.155	TELPAK A 3 CORE MODULES DEC RF08 + RS08 DISK UNIT PDP-9 + IDI 10000 + EX. ARITH.	3, 3

Table 6-25

Network Analysis, Three Users

MONTHLY COST, DOLLARS	AVERAGE RESPONSE TIME, SECONDS	EQUIPMENT COMPLEMENT	
7974	0.093	TELPAK A 3 CORE MODULES DEC RF08 + RS08 DISK UNIT PDP-9 + IDI 10000 + EX. ARITH.	3, 3

Table 6-25 (continued)

Network Analysis, Three Users

Figure 6-4

Minimum Response Times, Network Analysis

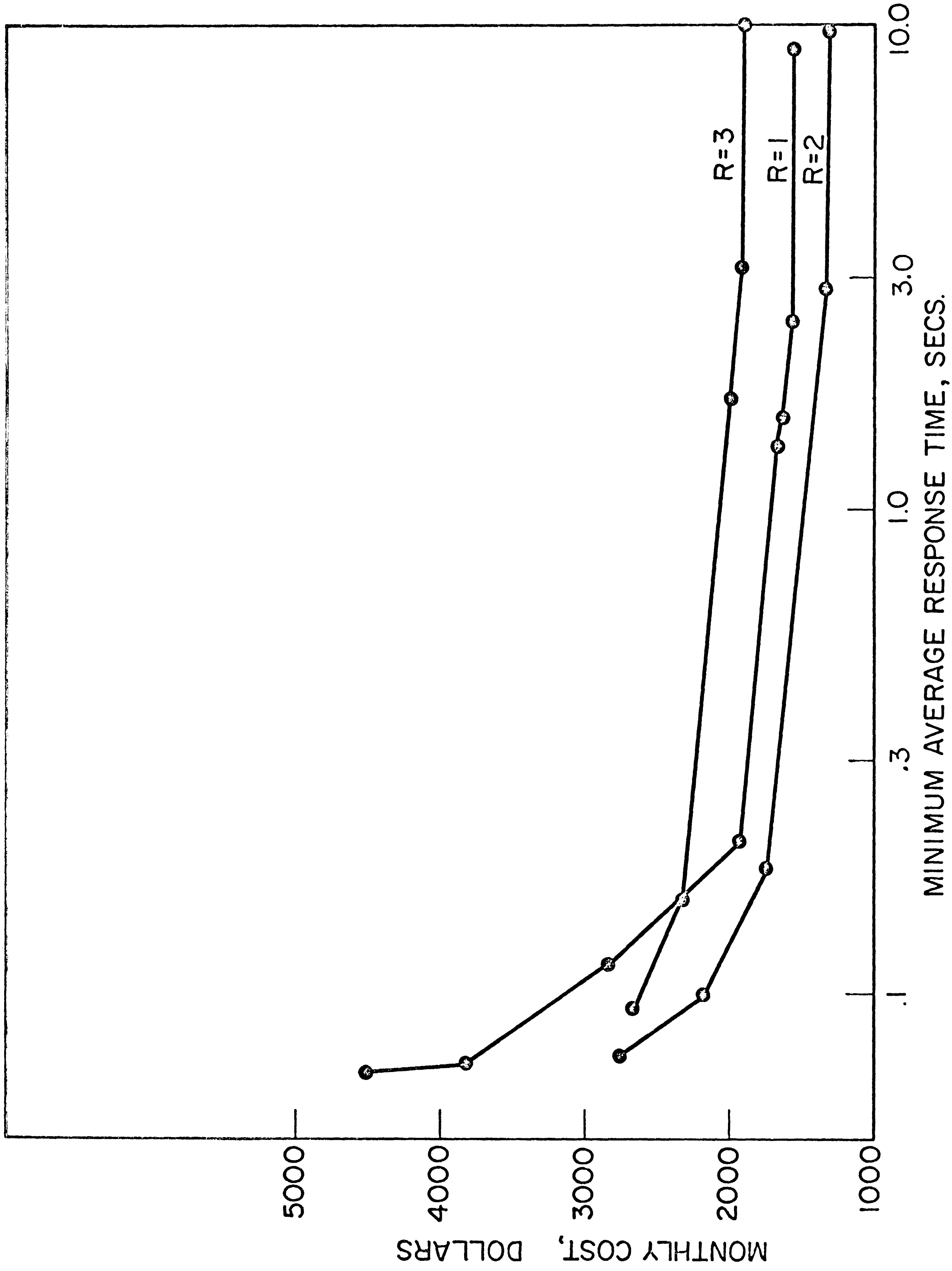

MONTHLY COST, DOLLARS
1000
2000
3000
4000
5000
MINIMUM AVERAGE RESPONSE TIME, SECS.
.1
.3
1.0
3.0
10.0
R=3
R=1
R=2

Figure 6-4

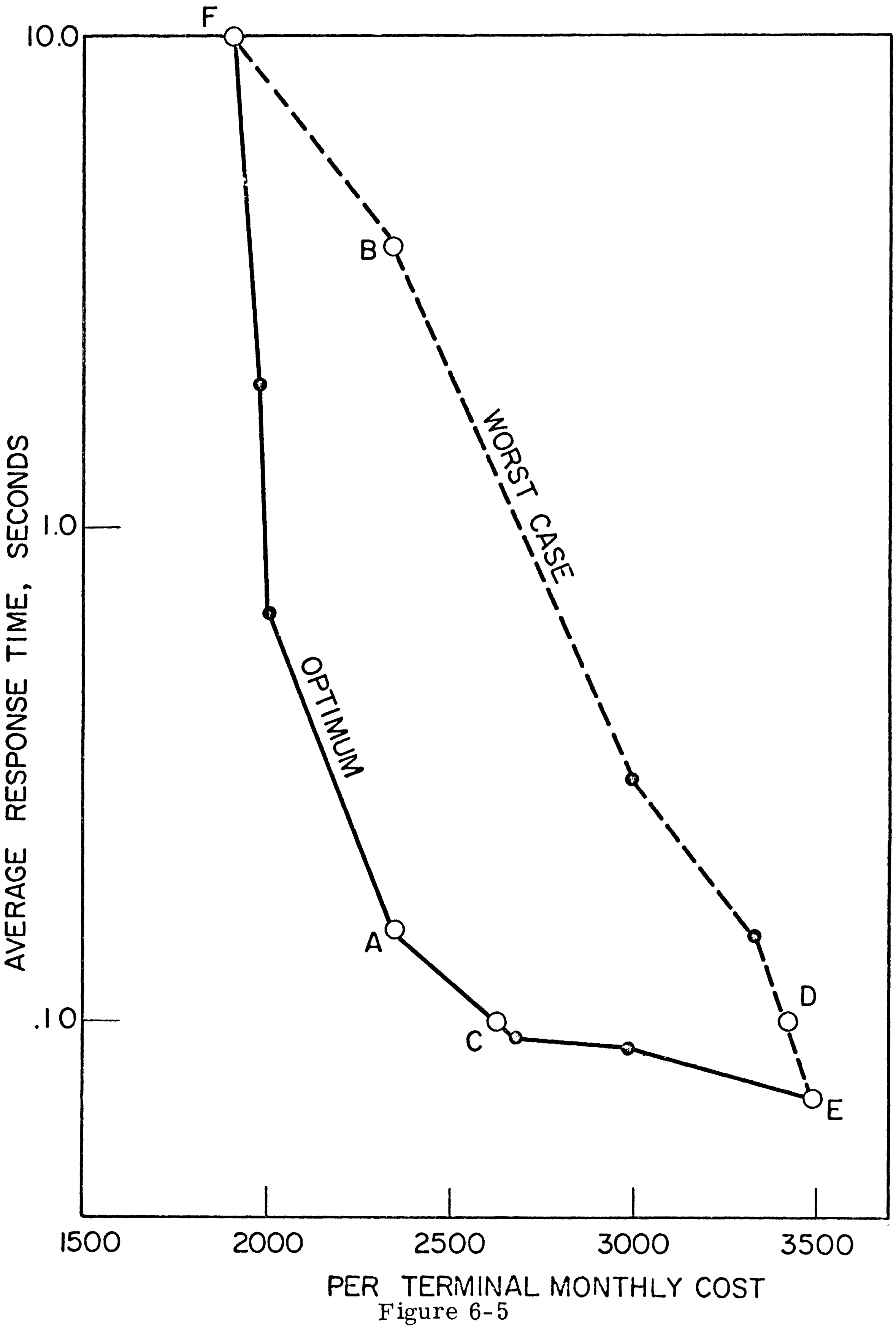
response time for each application implemented with a one, two, or three display console system.

6.4 Comparison of Best and Worst Display Systems

One of the purposes of this chapter, as stated in the opening discussion, is to demonstrate the usefulness and necessity of providing tools and guidelines for display system designers. It was stated that the difference in response times of optimal and non-optimal systems can be quite significant. Therefore, display systems should be designed using methods such as those presented in the preceding chapters, the alternative being poor response time for the number of dollars invested. Figure 6-5 shows just how severe the penalties of using a nonoptimal design can be. The figure plots response time versus cost for the network analysis application with three users. Both optimum and worst case response times for various values of cost are plotted. The worst case response time is the maximum response time of those display systems entered into the set S' during the optimization (Section4.2). It may be helpful to recall that S' contains only those display systems which, on the basis of their cost, are expected to provide good response time. There usually exists, however, display systems not in S' (therefore costing less than those in S') which can give better response time than the worst case response time. The graph's interpretation is that, for instance, if $2340 per month is

Figure 6-5

Best and Worst Average Response Times



F
10.0
B
WORST CASE
1.0
OPTIMUM
A
D
.10
C
E
1500
2000
2500
3000
3500
PER TERMINAL MONTHLY COST
AVERAGE RESPONSE TIME, SECONDS

Figure 6-5

to be spent on a display system, the response time can range from .155 seconds (Point A) to 3.92 seconds (Point B). These times differ by a factor of 25. Also, the graph can be interpreted to show that if a response time of .1 seconds is needed, the display system's monthly cost can vary from about \$2600 (Point C) to about \$3400 (Point D). This represents an unnecessary expenditure of up to \$800 per month!

There are other interesting aspects of the graph. Any display system selected from the set S' will fall within the bounded area in the graph, no matter what C'_{max} might have been used to find S'. In addition, the worst case curve and optimum curve meet at points E and F, because for their costs there was only one display system in S', so the one system gives at once the optimum and worst case response times.

In conclusion, the differences between optimum and worst case display systems are significant, with respect to both cost and response time. Therefore, display system designers must have at their disposal means of making intelligent design decisions, because the consequences of making bad decisions are too serious.

6.5 Interpretation of Results

The results presented in Section 6.2 can be used in several ways: as cost-effectiveness information, to tell when multiple-console systems should be used, and in the formulation of design guidelines. These three will be treated in turn.

6.5.1 Cost-Effectiveness

Any of the individual curves in Figures 6-1, 6-2, 6-3, and 6-4 can be interpreted from a cost-effectiveness viewpoint by a simple transformation. Let cost-effectiveness be defined as the number of user-display interactions/second obtained for each dollar spent. Then

$$\text{cost-effectiveness} = CE = \frac{\dfrac{1}{TMIN + 1/ARRIVE}}{C(\underline{X})/R} . \qquad (6.1)$$

TMIN + 1/ARRIVE is the time in seconds needed to complete one interaction; its reciprocal is the number of interactions per second. $C(\underline{X})$, defined in Section 2.6 is the display system's cost, and is divided by the number of users R to place the calculations on a per-user basis.

The results of applying equation 6.1 to the optimum display systems found for the 3-D drawing application (Figure 6-3) with one, two, and three display consoles are shown in Figure 6-6. The peaks in cost-effectiveness determine which of several optimum (in the sense of minimum response time) display systems is optimum in the cost-effectiveness sense. The three most cost-effective systems occur at costs of $2180, $2065, and $1999 per display console. This is again a reflection of economies of scale: for more users, the most effective system costs less, and provides more interactions/second/dollar. In Tables 6-14 to 6-25, an asterisk denotes the most cost-effective display systems.

Figure 6-6

Cost-Effectiveness

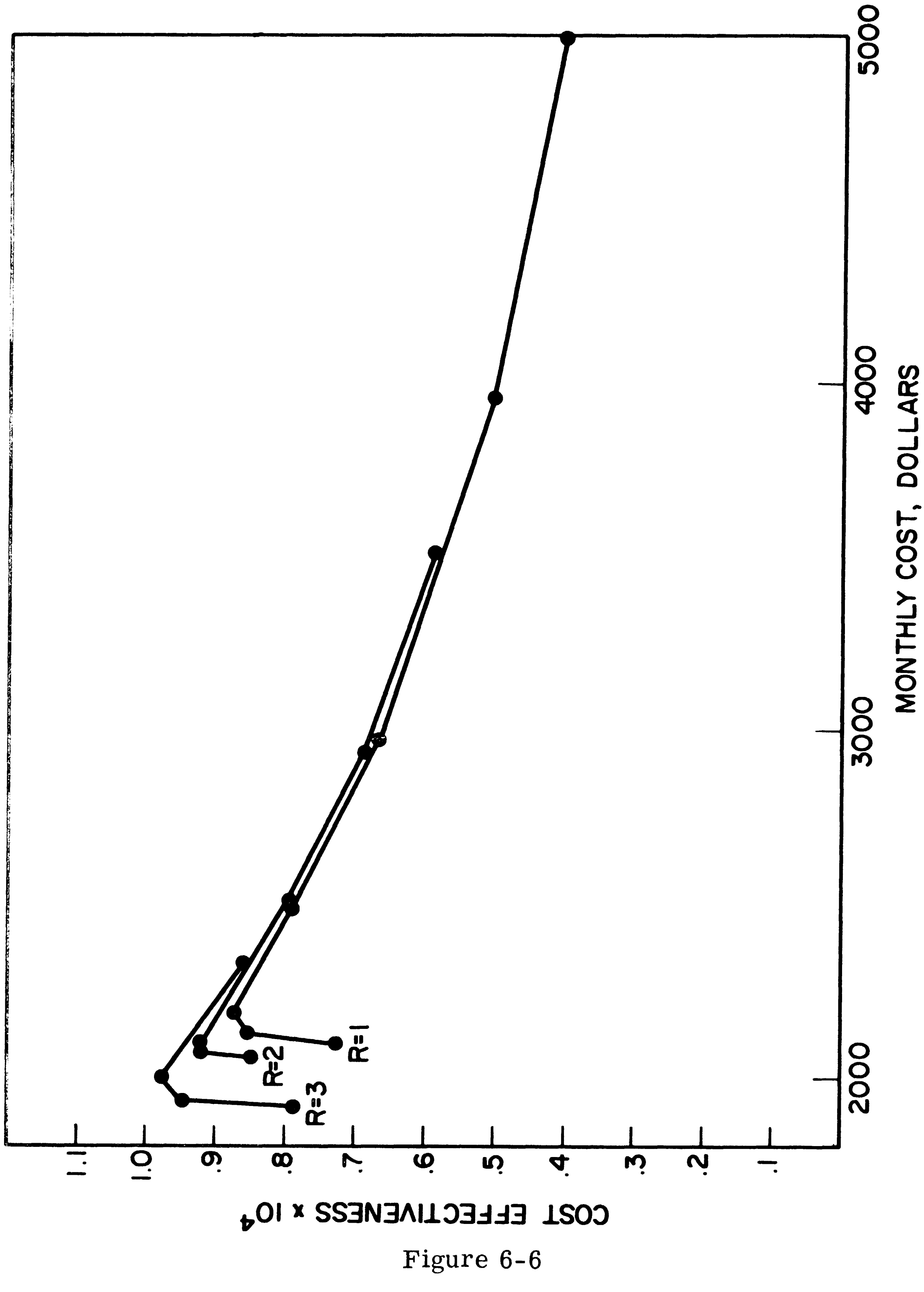

COST EFFECTIVENESS x 10^4
.1
.2
.3
.4
.5
.6
.7
.8
.9
1.0
1.1
MONTHLY COST, DOLLARS
2000
3000
4000
5000
R=1
R=2
R=3

Figure 6-6

A more sophisticated cost-effectiveness analysis would also take into account the monthly cost of the system's user, which would probably fall between \$2000 and \$3000. A display system judged best on the basis of equation 6. 1 and having a response time of 30 seconds wastes a lot of human time which, if taken into consideration, would force selection of a faster and more expensive display system.

On a more intuitive basis, Figures 6-1 to 6-4 all show that starting with the slowest responding systems, very small incremental expenditures (less than \$200) result in order-of-magnitude improvements in response time. As more and more money is invested, however, the returns become less and less significant. Indeed, there exists on many of the graphs a rather distinct breakpoint at which the slopes of the minimum response time curves become distinctly negative. This emphatically signals a diminishing returns situation, in which more expensive display systems give back relatively little in return for their increasing prices. The curves seem to say that it is wise to spend somewhat more than the necessary minimum, but not too much more. What should be purchased beyond a minimum system is discussed in Section 6. 5. 3. Before that, however, other information will be extracted from the results.

6.5.2 Multiple Versus Single Console Systems

When a display application is to be implemented for several simultaneous users, a systems design problem is created. If R display consoles are needed, should R separate display systems be obtained, or should a display system have multiple consoles? The immediate answer might be that economies of scale and sharing the display subsystems among several users should favor using multiple consoles. In most instances this is indeed the case. In the network analysis application, however, this is not true. As can be seen by examining Figure 6-4, the two console system is least expensive, on a per-console basis. Both the one and two console systems are less expensive than the three console system, except for response times less than .15 seconds. Although the three console system would be expected to be most economical, it is not.

The explanation for this is very simple. For the particular application considered here, only the most expensive remote computer-display controls were suitable for a three console system. Considerably less expensive units were useable for the one and two console systems, but because of the application's display requirements, they were not extendable to the three console system. For the other applications, the display requirements were such that this situation never arose.

The point to be remembered here is that the obvious is not always true: a single three console display system is not necessarily less expensive than three display systems each with one console. This can be so because lack of appropriate equipment hampers realization of normally intrinsic economies of scale.

6.5.3 Guidelines

General guidelines for display system designers are like Homer's Sirens: very desirable and very dangerous. They are desirable to help the designer carry out his work; they are dangerous because they can be improperly used and impart to the designer a false sense of security. While it is not necessary to deafen the crew to resist their temptations, design guidelines do need to be approached with a word of caution.

Guidelines are not absolutes, they are aids to be used in context. That is, guidelines can be used only within the context of applications for which they are intended. The applications studied here have been selected to provide a reasonably broad base on which to establish guidelines, so the guideline's usefulness can go across (but not necessarily beyond) the base. The implicit danger with guidelines is that it is easy to use them where they do not apply: it is easy for a designer to become little more than a cookbook technician, merely looking up guidelines and indiscriminately applying them. Having given this warning, the optimization results in Tables 6-14 to 6-25 will be examined in an attempt to deduce general guidelines for display system designers.

The most striking phenomenon noticed in these tables is that as additional money is spent, the first subsystem to be upgraded is the data link. This is because small amounts of additional money yield large improvements in data link transmission rate, which are in turn reflected in significant improvements in average response time. Note that this might not be the case with applications making less use of the data link than the applications studied here.

Once the data link has been upgraded to the synchronous switched line or private voice grade line level (2000 or 2400 bps), additional speed increases become considerably more expensive, as seen from Table 6-1. For the applications being studied, small incremental amounts of money must therefore either go toward more core memory, bulk storage, or a faster remote computer-display control. In every instance bulk storage is added, and in some cases the computer-display control is improved (only in cases when an improvement is inexpensive). Only twice is core memory added. In some cases, the data link speed is decreased, because adding bulk storage decreases data link traffic and thus makes a fast data link less important. This is particularly evident in the text editing (Tables 6-14 to 6-16) and the two-dimensional drawing (Tables 6-17 to 6-19) applications, for which response times on the order of 80 milliseconds are obtained with data links no faster than private voice grade lines. For three-dimensional drawing, about a 100 millisecond response is obtained, and for network analysis, about 200 milliseconds. The next step beyond these response times is taken with Telpak A service, providing a transmission rate of 40, 000 bps, and also providing a substantial increase in cost. By consulting the graphs of Figures 6-1 to 6-4 it is seen that the response times mentioned above correspond in general to points beyond which the returns on expenditures for Telpak A and other equipment decrease significantly. However, if better response time is needed, the results show that both Telpak A service and additional bulk storage are the most rewarding; significantly increasing core storage and remote computer-display control capability are relatively less important.

The general guidelines might thus read as follows: "A satisfactory inexpensive display system uses a voice grade data link, no bulk storage, little or no core storage beyond the absolute minimum needed, and the least expensive remote computer-display control. For little additional expenditure, the addition of a significant amount of bulk storage provides better response time. Inexpensive increases in the remote computer-display control's capabilities are also helpful. Further response time decreases are achieved with broad band data link speeds and more bulk storage. Additional response time improvements are obtained (at high cost) first by improving the remote computer-display control and then by using more core storage."

6.5.4 Hardware Aids for Multiplication and Division

The use of a hardware matrix multiplier for rotation of objects in the three-dimensional drawing application doesn't seem to be worth the projected price ($10,000 or $250 per month). For one, two, and three users, addition of the matrix multiplier (for one user, as part of the AGT-50) occurs only after most of the other display subsystems are at or near their maximum capability (Tables 6-20 to 6-22). The improvement in average response time ranges from 1 to 4 milliseconds - small indeed.

The use of hardware multiply-divide facilities, however, is another matter. For the three applications (all but text editing) which make use of fast multiplication and/or division, this capability is usually added at the level of expenditure where the general guidelines require the addition of bulk storage. This is just one of the inexpensive improvements in the remote computer-display control called for by the guidelines.

6.6 Division of Processing

In Section 2.7.1 one means for dividing processing of various interaction types between the main and remote computers was devised. It is desirable to answer two questions concerning this area:

1. Is the division of processing method better than the less sophisticated alternative of using a predetermined fixed division no matter how the display system be configured?
2. Are the results reasonable?

An experiment was designed to answer the first question. Using an application virtually identical to two dimensional drawing, an optimization was performed. During the course of the optimization, the division of processing algorithm produced several different values of PM, the display terminal's dependence on the main computer (Section 2.4). Among these values were 0.0, 0.055, and 0.105. The optimum display system occurred for $\underline{X} = (1, 3, 7, 4)$ with PM = 0.0. Next, optimizations were performed with a display system model using a fixed processing division, one optimization for each of the three PM's mentioned. The results were for PM = 0.0, $\underline{X} = (1, 3, 7, 4)$, $T = .02$; PM = 0.055, $\underline{X} = (6, 1, 6, 3)$, $T = .04$; and PM = 0.105, $\underline{X} = (6, 1, 6, 3)$, $T = .05$. The first important result here is that the $\underline{X}$'s are not all the same. If they had been, there would be little justification for calculating the optimum division of processing, as the results would be independent (at least in this instance) of the exact division. The second

	One User	Two Users	Three Users
Text Editing	.0	.0	.0
Two Dimensional Drawing	.075	.085	.075
Three Dimensional Drawing	.28	.125	.005
Network Analysis	.324	.324	.324

Table 6-26

PM for the Most Cost-Effective Display Systems important result is that the smallest T occurred for PM = 0.0, showing that the optimum division really does minimize response time.

The second question can be answered by examing Tables 6-7 to 6-10. The three rightmost columns show, by using M for main computer and R for remote computer, the division of processing for the display systems chosen in Section 6.5.1 as most cost-effective. These tables show that the relatively more demanding interaction types are processed by the main computer, while the remaining types are processed by the remote computer. This is as it should be. To complete the picture, Table 6-26 gives PM for each of the twelve systems.

6.7 Summary

The goals set forth at the start of this chapter have been met. The techniques developed in earlier chapters have been applied to several display applications, and the results have been interpreted so as to justify the search for display system design guidelines. These guidelines have been stated. Of course these guidelines must be used in the proper perspective, first because parameters on which they are based were only estimated, and second because the relative costs of the subsystems studied might change in the future. The next and concluding chapter will present an overview of what has been done, and will attempt to place the work reported here in the proper perspective.

Chapter VII

CONCLUSIONS

7.1 Review of the Research

The purpose of this research has been to develop aids for the analysis and design of highly interactive graphical display systems. These aids are needed because inept design decisions can be very wasteful.

To provide a framework within which to work, a mathematical model of a display system was developed in Chapter II. The model was a description of the system's application and hardware to predict response time. So that the model would be simple enough to facilitate quick analysis, and to make it realistic to use, the level of detail is relatively crude.

Along with the model, a method was proposed for dividing display processing between the main and remote computers so as to help minimize response time. This method is an aid for making one of the more important design decisions associated with display systems, namely, how much should the main computer do, and how much should the remote computer do?

Chapter III was devoted to a discussion of the relative merits of two analysis methods which could be used with the display model, simulation and numerical queueing analysis. It was shown that numerical queueing analysis is much faster (in terms of computer time) than simulation, and produces equally satisfactory results.

In Chapter IV an optimization technique was developed. It very efficiently finds an optimum display system having no more than a given cost. This optimization, which in turn uses the display system model, is the primary analytical tool developed in this research. All else is subsidiary to it. The optimization can be used by display system designers as a tool in itself. It is used here to study several display applications, and to find design guidelines.

Chapter V presents a methodical procedure for ranking various remote computer-display controls on the basis of their display and computational capabilities. This procedure is useful to display system designers in and of itself, and also is used here with the display system model.

The usefulness of all this work becomes apparent in Chapter VI. The characteristics of four different display system applications are estimated (not measured), and many optimum display systems are found. These results are used to demonstrate the necessity for system design aids, and to develop some guidelines for system designers.

7.2 Critical Evaluation

What are the good points of this research? First and foremost, it provides system analysts and designers with a conceptual basis on which their efforts can be established. The steps required for designing display systems on an objective basis have been illustrated. Second, the specific mathematical tools developed here have been shown to be useful. Finally, the design guides should prove helpful to system designers.

What is wrong with the research? First, it is most difficult to evaluate the effect of the various simplifications and assumptions which have been made to develop the display system model. Second, the model's output is only as good as its inputs - the application and hardware specifications. Because the appropriate data is not available, the application specifications have been estimated. They may or may not be satisfactory. The taking of detailed data on display applications, rather than the general data available in the appendix, would be most helpful. As time goes by, costs of the display subsystems will change. The guidelines will therefore also change: they are not lasting and enduring, although the techniques used to find them are.

Despite these weaknesses, and because of the strong points, it is felt that this research is a solid beginning toward the Analysis and Design of Interactive Graphical Display Systems, on which others can build.

REFERENCES

[1] Adage, Inc. System Reference Manual - Adage Graphics Terminal, Boston, 1968.

[2] Adams Associates, The Computer Display Review, Bedford, Massachusetts, 1968.

[3] Adams, Charles W., "Grosch's Law Repealed," Datamation 8, 7 (July 1962), pp. 38-39.

[4] Annerman, Anne B. et.al., Displaytran - A Graphical Display Oriented Conversational Fortran Facility for an IBM 360/40 Computer, Technical Memo. No. K-39/67, U.S. Naval Weapons Laboratory, Dahlgren, Virginia, July 1967.

[5] Arbuckle, R. A., "Computer Analysis and Throughput Evaluation," Computers and Automation 15, 1 (Jan. 1966), pp. 12-15, 19.

[6] Ball, N. A. et.al., "A Shared Memory Computer Display System," IEEE Trans. on Elec. Comp. 15, 5 (Oct. 1966), pp. 750-756.

[7] Corbato, F. J. and V. A. Vyssotsky, Introduction and Overview of the Multics System, Proc. FJCC, Spartan Books, Washington D. C., 1965, pp. 185-196.

[8] Corbato, F. J. et.al., An Experimental Time-Sharing System, Proc. SJCC, National Press, Palo Alto, California, 1962, pp. 335-344.

[9] Cross, P., "A Logic Drawing Board for the PDP7/340," The Computer Bulletin 11, 3 (Dec. 1967), pp. 237-245.

[10] Cox, D. R. and Walter L. Smith, Queues, John Wiley and Sons, Inc., New York, 1961.

[11] David, M. R. and T. O. Ellis, The RAND Tablet; A Man-Machine Graphical Communication Device, Proc. FJCC, Spartan Books, Washington D. C., 1964, pp. 325-331.

[12] Dawson, D. F. et.al., "Computer-Aided Design of Electronic Circuits - A User's Viewpoint," Proc. IEEE 55, 11 (Nov. 1967), pp. 1946-1954.

[13] Dertouzos, Michael L. "An Introduction to On-Line Circuit Design," Proc. IEEE 55, 11(Nov. 1967), pp. 1961-1971.

[14] Dertouzos, Michael L., "CIRCAL: On-Line Circuit Design," Proc. IEEE 55, 5 (May 1967), pp. 637-654.

[15] Digital Equipment Corporation, PDP-8 User Handbook, Publication F-95, Maynard, Massachusetts, 1968.

[16] Digital Equipment Corporation, Precision Incremental CRT Display Type 340, Maynard, Massachusetts.

[17] Digital Equipment Corporation, Small Computer Handbook, Publication C-800, Maynard, Massachusetts, 1968.

[18] Digital Equipment Corporation, 338 Buffered Display Instruction Manual, Publication DEC-08-H6AA-D, Maynard, Massachusetts.

[19] Display Techniques Branch, Rome Air Development Center, Compendium of Visual Displays (Second Revision), Rome Air Development Center, Rome, New York, 1967.

[20] Foley, James D., "A Markovian Model of the University of Michigan Executive System," Comm. ACM 10, 9 (Sept. 1967), pp. 584-588.

[21] Gruenberger, Fred, "Are Small Free-Standing Computers Here to Stay?," Datamation 12, 4 (April 1966), pp. 67-68.

[22] Gruenberger, Fred et.al., Computer Graphics, Thompson Book Co., Washington, D. C., 1967.

[23] Hargreaves, et.al., Image Processing Hardware for a Man-Machine Graphical Communication System, Proc. FJCC, Spartan Books, Washington, D. C., 1966, pp. 363-386.

[24] Hillegass, John R., "Standardized Benchmark Problems Measure Computer Performance," Computers and Automation 15, 1 (Jan. 1966), pp. 16-19.

[25] Information Displays, Inc., Modular Graphic CRT Displays Available from Information Displays, Inc., IDI Data Sheet 127-367, Mount Kisco, New York, 1967.

[26] Information Displays, Inc., The New IDIIOM, IDI Data Sheet 148-767, Mount Kisco, New York, 1967.

[27] International Business Machines Corporation, General Purpose Simulation System/360 User's Manual, IBM Publication H20-0326, White Plains, New York.

[28] International Business Machines Corporation, S/360 General Program Library, GPAK, An Online System/360 Graphic Data Processing Subroutine Package with (V1M1) Realtime 2250 Input and Display, White Plains, New York.

29] International Business Machines Corporation, "1620 Electronic Circuit Analysis Program (ECAP)," User Manual, IBM Report 1620-EE-02X, White Plains, New York.

[30] Joslin, Edward O. and John J. Aiken, "The Validity of Basing Computer Selections on Benchmark Results," Computers and Automation 15, 1 (Jan. 1966), pp. 22-23.

[31] Karplus, Walter J. (ed.), On Line Computing, McGraw-Hill, New York, 1967.

[32] Kennedy, James R., A System for Time-Sharing Graphic Consoles, Proc. FJCC, Spartan Books, Washington, D. C., 1966, pp. 211-222.

[33] Knight, Kenneth E., A Fast Sort of Country, Unpublished Ph. D. Dissertation, Graduate School of Industrial Administration, Carnegie Institute of Technology, Pittsburgh, Pennsylvania.

[34] Knight, Kenneth E., "Changes in Computer Performance," Datamation 12, 9 (Sept. 1966), pp. 40-57.

[35] Knight, Kenneth E., "Evolving Computer Performance, 1963-1967," Datamation 14, 1 (Jan. 1968), pp. 31-35.

[36] Koford, J. S. et. al., Using a Graphic Data-Processing System to Design Artwork for Manufacturing Hybrid Integrated Circuits, Proc. FJCC, Spartan Books, Washington, D. C., 1966, pp. 229-246.

[37] Konkle, K. H., "An Analog Comparator as a Pseudo-Light Pen for Computer Displays," IEEE Trans. on Elec. Comp. 17, 1 (Jan. 1968), pp. 54-55.

[38] Kucera, John J., "Transfer Rate of Information Bits," Computer Design 7, 6 (June 1968), pp. 56-69.

[39] Lathrop, Jay W. et. al., "A Discretionary Wiring System as the Interface Between Design Automation and Semiconductor Manufacture," Proc. IEEE 55, 11 (Nov. 1967), pp. 1988-1997.

[40] Laver, Hugh C., Bulk Core in a 360/67 Time-Sharing System, Proc. FJCC, Thompson Books, Washington, D. C., 1967, pp. 601-609.

[41] Lawler, E. L. and M. D. Bell, "A Method for Solving Discrete Optimization Problems," Operations Research 14, 6 (Nov. - Dec. 1966), pp. 1098-1112.

[42] Lourie, Janice R. and John J. Lorenzo, Textile Graphics applied to Textile Printing, Proc. FJCC, Thompson Books, Washington, D. C., 1967, pp. 33-40.

[43] Nielsen, Norman R., "The Simulation of Time Sharing Systems," Comm. ACM 10, 7 (July 1967), pp. 397-412.

[44] Parzen, Emanuel, Stochastic Processes, San Francisco, Holden-Day, 1965.

[45] Prince, David M., "Man-Computer Graphics for Computer-Aided Design," Proc. IEEE 54, 12 (Dec. 1966), pp. 1698-1708.

[46] Pryor, T. Allen and Homer R. Warner, "Time Sharing in Biomedical Research," Datamation 12, 4 (April 1966), pp. 54-63.

[47] Ruiz-Pala, E. et.al., Waiting Line Models, Reinhold Publishing Corporation, New York, 1967.

[48] Schwartz, J. I., E. G. Coffman, and C. Weissman, A General Purpose Time-Sharing System, Proc. FJCC, Spartan Books, Baltimore, 1964, pp. 397-411.

[49] Schubik, Martin, "Simulation of the Industry and the Firm," American Economic Review L, 5 (Dec. 1960), pp. 908-919.

[50] So, Hing C., "OLCA: An On-Line Circuit Analysis System," Proc. IEEE 55, 11 (Nov. 1967), pp. 1954-1961.

[51] Solomon, Martin B., Jr., "Economies of Scale and the IBM System/360," Communications of the ACM 9, 6 (June 1966), pp. 435-440.

[52] Spitalny, Arnold and Martin J. Goldberg, "On-Line Graphics Applied to Layout Design of Integrated Circuits," Proc. IEEE 55, 11 (Nov. 1967), pp. 1982-1988.

[53] Stotz, R. H. and J. E. Ward, Operating Manual for the ESL Console, ESL Internal Memorandum 9442-M-129, MAC Internal Memorandum MAC-M-217, Massachusetts Institute of Technology.

[54] Sutherland, Ivan E., Sketchpad, A Man-Machine Graphical Communication System, Proc. SJCC, Spartan Books, Washington, D. C., 1963, pp. 329-346.

[55] Teichroew, Daniel and John F. Lubin, "Computer Simulation - Discussion of the Technique and Comparison of Languages," Communications of the ACM 9, 10 (Oct. 1966), pp. 723-741.

[56] Temes, Gabor C. and Donald A. Calahan, "Computer-Aided Network Optimization - The State-of-the-Art," Proc. IEEE 55, 11(Nov. 1967), pp. 1832-1863.

[57] University of Michigan Computing Center, Michigan Terminal System, (2nd Ed.), Ann Arbor, Michigan, 1967.

[58] Vorhaus, A. H., "General Purpose Display System," Datamation 12, 7 (July 1966), pp. 59-64.

[59] Wallace, Victor L. and Richard S. Rosenberg, Markovian Models and Numberical Analysis of Computer System Behavior, Proc. SJCC, Spartan Books, Washington, D. C., 1966, pp. 141-148.

[60] Wallace, Victor L. and Richard S. Rosenberg, RQA-1, The Recursive Queue Analyzer, Systems Engineering Laboratory Technical Report no. 2, The University of Michigan, Ann Arbor, Michigan, 1966.

[61] Ward, John E., "Systems Engineering in Computer-Driven CRT Displays for Man-Machine Communication," IEEE Trans. on Systems Science and Cybernetics 3, 1 (June 1967), pp. 47-54.

[62] Williams, T. G., Time Sharing System Organization for Computer Graphics, Proc. Second Hawaii International Conference on System Sciences, Western Periodicals Company, 1969.

[63] Wood, L. H. et. al., "The Amtram Input-Output Terminal," Computer Design 7, 3 (March 1968), pp. 68-74.

Appendix A

THE PREPROCESSOR PROGRAM, AND ITS INPUT DATA

The purpose of this appendix is to discuss the PREPROCESSOR program, and its required inputs. This program is used to determine which of many remote computer-display controls are appropriate for a display application. The decision is based on each unit's cost, instruction execution rate, and display capability, as was outlined in Figure 4-8. In the program the reciprocal of X4, instruction execution rate, is used in place of X4, and is called T. Therefore certain of the inequalities in Figure 4. 8 are reversed in the program. Naturally, only units with the required number of display consoles are considered.

There are two sets of input data for PREPROCESSOR. The first, loosely called SOFTWAREDATA, describes the display application to be implemented. These parameters are R, the number of required display consoles; Q_{min}, the minimum acceptable quantity of display material; x, the fraction of displayed information shared among all consoles; Ω_i, $i=1, 2, \ldots, 5$, the weights to be applied to the five standard display test patterns; and w_i, $i=1, 2, \ldots, 42$, the relative usage of each display instruction. An example of this data is shown in Figure A-1.

A second set of data, called HARDWAREDATA, describes all the remote computer-display controls which are being considered.

Each remote computer can be used in up to eight different display control-display console configurations, as in Table 6-5. Accordingly, the description of each remote computer-display control consists of first an alphanumeric description of the unit, and quantities called CRT(I, K) and CNT(I, K), K=1, 2, . . . , 8, which are the number of display consoles and controls in each of the eight possible configurations. I is used to index the various remote computer-display control types. Next is the total dollar cost of each configuration (without any core memory), and finally, the time (in microseconds) taken by this unit to execute each of the 42 display instructions. The complete remote computer-display control data base can be seen in Figure A-2. Aid in interpreting it can be had by examining the READ statements and their FORMATS in Figure A-3, which is a listing of the PREPROCESSOR program. HARDWAREDATA and SOFTWAREDATA input is via unit 5, and output data is on unit 6. A typical output of the program is shown by Table 6-6. The two digits following the unit's alphanumeric description are the number of display controls and display consoles, respectively.

Figure A-1

Description of Display Applications

```
$LIST SOFTWAREDATA
  1     DESCRIPTION OF GENERAL 2-DIMENSIONAL DRAWING APPLICATION
  2     1
  3     0075
  4     1.00000
  5     .0   .0   .0   .5   .5
  6     .01  .01  .01  .01  .0025.0025.0025.0025.0025.52            .001
  7               .015 .015 .005      .005 .005 .02  .02  .02  .065 .065 .065
  8     .01  .01  .01            .005      .0765.01  .01
  9     DESCRIPTION OF A GENERAL 3-DIMENSIONAL DRAWING APPLICATION
 10     1
 11     0150
 12     1.0
 13     .0   .0   .0   .5   .5
 14     .0063.0063.0063.0063.0015.0015.0015.0015.0015.3245.0625.0007
 15               .0094.0094.0031     .0031.0031.0126.0126.0126.0413.0413.0413
 16     .0063.0063.0063          .0031     .0478.0063.0063.0781.0781.0781.0781
 17     DESCRIPTION OF A GENERAL NETWORK ANALYSIS PROGRAM
 18     1
 19     0060
 20     1.0
 21     .1000.0000.2000.0000.7000
 22     .0067.0067.0067.0067.0016.0016.0016.0016.0016.3476     .0007
 23               .01  .01  .0033     .0033.0033.0134.0134.0134.044 .044 .044
 24     .0067.0067.0067          .0033     .051 .0067.0067.0833.0833.0833.0833
 25     DESCRIPTION OF A TEXT-EDITING APPLICATION
 26     1
 27     0100
 28     1.0
 29     .9999.0000.0000.0000.0000
 30     .01  .01  .01  .01  .01  .01            .02  .1
 31               .05  .05            .025 .025 .025 .025      .1   .1   .1
 32     .1   .01  .01            .1        .03  .03  .04
```

Figure A-2

Description of Remote Computer-Display Controls

```
$LIST HARDWAREDATA
    1     42
    1.1   039
    1.5   DEC 338 REMOTE DISPLAY TERMINAL
    2     11213141224233344
    3          48000     62700     75400     88100     85000    114400    122000    159000
    4              115         216          68         120          51
    5         26     8    15     5    16    16    14    14    29   450  1160    28    11    15
    6          0     0    14    15   275   270    11    13    11    15     5    31    35    40
    7         25    25    25    25    90    16   142    10     8    30   192   192   900   900
    8     DEC 338 + EXTENDED ARITHMATIC
    9     11213141224233344
    10         51500     66200     79900     91600     88500    117900    125500    162500
    11             115         216          68         120          51
    12        26     8    15     5    16    16    14    14    29   450   175    28    11    15
    13                    14    15    40    40    11    13    11    15     5    31    35    40
    14        25    25    25    25    90    16   142    10     8    30   192   192   150   150
    15    DEC 339
    16    11213141224233344
    17         55000     69700     82400     95100     92000    121400    129000    166000
    18             115         216          68         120          51
    19        18     6    10     5    10    10     7     7    17   250   652    18     5     8
    20                    18    19    21    21     6     7     6     8     3    21    30    28
    21        16    16    16    16    56    10    83     4     4    27   120   120   700   700
    22    DEC 339 + EXTENDED ARITHMATIC
    23    11213141224233344
    24         59000     73700     86400     99100     96000    125400    133000    170000
    25             115         216          68         120          51
    26        18     6    10     5    10    10     7     7    17   250    89    18     5     8
    27                    18    19    21    21     6     7     6     8     3    21    30    28
    28        16    16    16    16    56    10    83     4     4    27   120   120   100   100
    29    PDP-8 + DEC 340 DISPLAY
    30    11213141224233344
    31         41000     58700     71400     84100     71000    106400    101000    131000
    32              84         204          64         100          52
    33     00026     8    13     7   145   160    14    14    13   500  1565   110    70    15
    34        75    80    14    15   275   270    11    13    11    15     5    31    35    40
    35        25    25    25    25    90    16   142    10     8    30   192   192   900   900
    36    PDP-8 + 340 DISPLAY + EX. ARITH.
    37    11213141224233344
    38         44500     62200     74900     87600     74500    109900    104500    134500
    39              84         204          64         100          52
    40     00026     8    13     7   145   160    14    14    13   500   480   110    70    15
    41        75    80    14    15    40    40    11    13    11    15     5    31    35    40
    42        25    25    25    25    90    16   142    10     8    30   192   192   150   150
    43    PDP-9 + DEC 340 DISPLAY
    44    11213141224233344
    45         47000     64700     77400     90100     77000    112400    117000    147000
    46              84         204          64         100          52
    47        18     6     9     6    91   101     7     7     7   300   844    66    31     8
    48        56    55    18    19   150   150     6     7     6     8     3    21    30    28
    49        16    16    16    16    56    10    83     4     4    24   120   120   700   700
    50    PDP-9 + DEC 340 DISPLAY + EX. ARITH
    51    11213141224233344
    52         51000     68700     91400     94100     91000    116400    121000    151000
    53              84         204          64         100          52
    54        18     6     9     6    91   101     7     7     7   300   281    66    31     8
    55        56    55    18    19    21    21     6     7     6     8     3    21    30    28
    56        16    16    16    16    56    10    83     4     4    24   120   120   100   100
```

Figure A-2 (continued)

Description of Remote Computer-Display Controls

```
 57    DEC PDP-8 + IDI 10000
 58    1121314122423344
 59       76430   91930  107430  122900  141860  172860  207290  272721
 60           105        871        157        312        124
 61       26     8    15     5    16    16    14    14    29   450  1160    28    11    15
 62        0     0    14    15   275   270    11    13    11    15     5    31    35    40
 63       25    25    25    25    90    16   142    10     8    30   192   192   900   900
 64    PDP-8 + IDI 10000 + EXTENDED ARITH.
 65    1121314122423344
 66       79930   95430  110930  126400  145360  176360  210790  276221
 67           105        871        157        312        124
 68       26     8    15     5    16    16    14    14    29   450   175    28    11    15
 69                   14    15    40    40    11    13    11    15     5    31    35    40
 70       25    25    25    25    90    16   142    10     8    30   192   192   150   150
 71    DEC PDP-8 + IDI 11000
 72    1121314122423344
 73       68260   76410   84560   92710  124520  140820  180780  237040
 74           116        280         60        152         18
 75       26     8    15     5    16    16    14    14    29   450  1160    28    11    15
 76        0     0    14    15   275   270    11    13    11    15     5    31    35    40
 77       25    25    25    25    90    16   142    10     8    30   192   192   900   900
 78    PDP-8 + IDI 11000 + EXTENDED ARITH.
 79    1121314122423344
 80       71760   79910   88060   96210  126020  144320  184280  240540
 81           116        280         60        152         18
 82       26     8    15     5    16    16    14    14    29   450   175    28    11    15
 83                   14    15    40    40    11    13    11    15     5    31    35    40
 84       25    25    25    25    90    16   142    10     8    30   192   192   150   150
 85    PDP-9 + IDI 10000
 86    1121314122423344
 87       82500   98500  114500  130500  147000  177000  211500  286000
 88           105        871        157        312        124
 89       18     6    10     5    10    10     7     7    17   250   652    18     5     8
 90                   18    19    21    21     6     7     6     8     3    21    30    28
 91       16    16    16    16    56    10    83     4     4    27   120   120   700   700
 92    PDP-9 + IDI 10000 + EXTENDED ARITH.
 93    1121314122423344
 94       86500  102500  118500  134500  151000  181000  215500  290000
 95           105        871        157        312        124
 96       18     6    10     5    10    10     7     7    17   250    89    18     5     8
 97                   18    19    21    21     6     7     6     8     3    21    30    28
 98       16    16    16    16    56    10    83     4     4    27   120   120   100   100
 99    PDP-9 + IDI 11000
100    1121314122423344
101       74260   82410   90560   98710  130520  146820  186780  243040
102           116        280         60        152         18
103       18     6    10     5    10    10     7     7    17   250   652    18     5     8
104                   18    19    21    21     6     7     6     8     3    21    30    28
105       16    16    16    16    56    10    83     4     4    27   120   120   700   700
106    PDP-9 + IDI 11000 + EXTENDED ARITH.
107    1121314122423344
108       78260   86410   94560  102710  134520  150820  190780  247040
109           116        280         60        152         18
110       18     6    10     5    10    10     7     7    17   250    89    18     5     8
111                   18    19    21    21     6     7     6     8     3    21    30    28
112       16    16    16    16    56    10    83     4     4    27   120   120   100   100
113    IDI INPUT OUTPUT MACHINE (IDIIOM)
114    1121314100000000
115       71000   84300   97600  110900
116           109        781        150        310        129
```

Figure A-2 (continued)

Description of Remote Computer-Display Controls

```
117               14      9     13      2     11     11     13     14      7    250    750     22     14     14
118               50     50     14     14    180    180     14     14     14     14      3     27      4     30
119                7      7      7      7    205     16    249      5      7     14    200    200    900    900
120          IDIIOM + EXTENDED ARITHMATIC
121          1121314100000000
122               73500      86800     100100     113400
123                  109           781           150           310           129
124               14      9     13      2     11     11     13     14      7    250    102     22     14     14
125               50     50     14     14     18     27     14     14     14     14      3     27      4     30
126                7      7      7      7    205     16    249      5      7     14    200    200    172    270
127          ADAGE AGT 10
128          1121314100000000
129               50000      61300      72600      83900
130                  130           525           200           220            65
131               24      5      8            20     20     16     16     10    200    144     16     20      8
132               40     45     28     28     40     40     24     24     20     28      6     40     10     52
133              200     20     20     26    208     40    320     16     16     24    144    144    126    126
134          ADAGE AGT 30
135          1121314100000000
136               95000     106300     117600     128900
137                  130           525           200           220            65
138              101     11      8            20     20     16     16     10    200      1     16     20      8
139               40     45     28     28     40     40     24     24     20     28      6     40     10     52
140              200     20     20     26    208     40    320     16     16     24    144    144    126    126
141          ADAGE AGT 50
142          1121314100000000
143              125000     136300     147600     158900
144                  160           650           250           270            80
145              116      4      4            20     20     16     16     10    200      1     16     20      8
146               40     45     14     14     20     20     12     12     10     14      3     20      5     26
147              100     10     10     13    104     20    160      9      8     12     72     72     63     63
201          DEC 338 WITH MATRIX MPY
202          1121314122423344
203               58000      72700      85400      98100     105000     134400     152000     159000
204                  115           216            68           120            51
205               26      8     15      5     16     16     14     14     29    450      1     28     11     15
206                0      0     14     15    275    270     11     13     11     15      5     31     35     40
207               25     25     25     25     90     16    142     10      8     30    192    192    900    900
208          DEC 338 + EAE + MATRIX
209          1121314122423344
210               61500      76200      88900     101600     108500     137900     155500     202500
211                  115           216            68           120            51
212               26      8     15      5     16     16     14     14     29    450      1     28     11     15
213                             14     15     40     40     11     13     11     15      5     31     35     40
214               25     25     25     25     90     16    142     10      8     30    192    192    150    150
215          DEC 339 + MATRIX
216          1121314122423344
217               65000      79700      92400     105100     112000     141400     159000     206000
218                  115           216            68           120            51
219               18      6     10      5     10     10      7      7     17    250      1     18      5      8
220                             18     19     21     21      6      7      6      8      3     21     30     28
221               16     16     16     16     56     10     83      4      4     27    120    120    700    700
222          DEC 339 + EAE + MATRIX
223          1121314122423344
224               69000      83700      96400     109100     116000     145400     163000     210000
225                  115           216            68           120            51
226               18      6     10      5     10     10      7      7     17    250      1     18      5      8
227                             18     19     21     21      6      7      6      8      3     21     30     28
228               16     16     16     16     56     10     83      4      4     27    120    120    100    100
229          PDP-8 + DEC 340 + MATRIX
```

Figure A-2 (continued)

Description of Remote Computer-Display Controls

```
230     1121314122423344
231        51000   68700   81400   94100   91000  126400  131000  171000
232             84       204        64       100        52
233     00026    8   13    7  145  160   14   14   13  500    1  110   70   15
234        75   80   14   15  275  270   11   13   11   15    5   31   35   40
235        25   25   25   25   90   16  142   10    8   30  192  192  900  900
236     PDP-8 + DEC 340 + EAE + MATRIX
237     1121314122423344
238        54500   72200   84900   97600   94500  129900  134500  174500
239             84       204        64       100        52
240     00026    8   13    7  145  150   14   14   13  500    1  110   70   15
241        75   80   14   15   40   40   11   13   11   15    5   31   35   40
242        25   25   25   25   90   16  142   10    8   30  192  192  150  150
243     PDP-9 + DEC 340 + MATRIX
244     1121314122423344
245        57000   74700   87400  100100   97000  132400  147000  187000
246             84       204        64       100        52
247        18    6    9    6   91  101    7    7    7  300    1   66   31    8
248        56   55   18   19  150  150    6    7    6    8    3   21   30   28
249        16   16   16   16   56   10   83    4    4   24  120  120  700  700
250     PDP-9 + DEC 340 + EAE + MATRIX
251     1121314122423344
252        61000   78700   91400  104100  101000  136400  151000  191000
253             84       204        64       100        52
254        18    6    9    6   91  101    7    7    7  300    1   66   31    8
255        56   55   18   19   21   21    6    7    6    8    3   21   30   28
256        16   16   16   16   56   10   83    4    4   24  120  120  100  100
257     DEC PDP-8 + IDI 10000 + MATRIX
258     1121314122423344
259        86430  101930  117430  132900  161860  192860  237290  302721
260            105       871       157       312       124
261        26    8   15    5   16   16   14   14   29  450    1   28   11   15
262         0    0   14   15  275  270   11   13   11   15    5   31   35   40
263        25   25   25   25   90   16  142   10    8   30  192  192  900  900
264     PDP-8 + IDI 10000 + EAE + MATRIX
265     1121314122423344
266        89930  105430  120930  136400  165360  196360  240790  306221
267            105       871       157       312       124
268        26    8   15    5   16   16   14   14   29  450    1   28   11   15
269                  14   15   40   40   11   13   11   15    5   31   35   40
270        25   25   25   25   90   16  142   10    8   30  192  192  150  150
271     DEC PDP-8 + IDI 11000 + MATRIX
272     1121314122423344
273        78260   86410   94560  102710  144520  160820  210780  277040
274            116       280        60       152        18
275        26    8   15    5   16   16   14   14   29  450    1   28   11   15
276         0    0   14   15  275  270   11   13   11   15    5   31   35   40
277        25   25   25   25   90   16  142   10    8   30  192  192  900  900
278     PDP-8 + IDI 11000 + EAE + MATRIX
279     1121314122423344
280        81760   89910   98060  106210  146020  166320  214280  280540
281            116       280        60       152        18
282        26    8   15    5   16   16   14   14   29  450    1   28   11   15
283                  14   15   40   40   11   13   11   15    5   31   35   40
284        25   25   25   25   90   16  142   10    8   30  192  192  150  150
285     PDP-9 + IDI 10000 + MATRIX
286     1121314122423344
287        92500  108500  124500  140500  167600  197000  241500  326000
288            105       871       157       312       124
289        18    6   10    5   10   10    7    7   17  250    1   18    5    8
```

Figure A-2 (continued)

Description of Remote Computer-Display Controls

```
290                18    19    21    21     6     7     6     8     3    21    30    28
291    16    16    16    16    56    10    83     4     4    27   120   120   700   700
292   PDP-9 + IDI 10000 + EAE + MATRIX
293   1121314122423344
294       96500      112500      128500      144500      171000      201000      245500      330000
295         105         871         157         312         124
296    18     6    10     5    10    10     7     7    17   250     1    18     5     8
297                18    19    21    21     6     7     6     8     3    21    30    28
298    16    16    16    16    56    10    83     4     4    27   120   120   100   100
299   PDP-9 + IDI 11000 + MATRIX
300   1121314122423344
301       84260       92410      100560      108710      150520      166820      216780      283040
302         116         280          60         152          18
303    18     6    10     5    10    10     7     7    17   250     1    18     5     8
304                18    19    21    21     6     7     6     8     3    21    30    28
305    16    16    16    16    56    10    83     4     4    27   120   120   700   700
306   PDP-9 + IDI 11000 + EAE + MATRIX
307   1121314122423344
308       88260       96410      104560      112710      154520      170820      220780      287040
309         116         280          60         152          18
310    18     6    10     5    10    10     7     7    17   250     1    18     5     8
311                18    19    21    21     6     7     6     8     3    21    30    28
312    16    16    16    16    56    10    83     4     4    27   120   120   100   100
313   IDIIOM + MATRIX
314   1121314100000000
315       81000       94300      107600      120900
316         109         781         150         310         129
317    14     9    13     2    11    11    13    14     7   250     1    22    14    14
318    50    50    14    14   180   180    14    14    14    14     3    27     4    30
319     7     7     7     7   205    16   249     5     7    14   200   200   900   900
320   IDIIOM + EAE + MATRIX
321   1121314100000000
322       83500       96800      110100      123400
323         109         781         150         310         129
324    14     9    13     2    11    11    13    14     7   250     1    22    14    14
325    50    50    14    14    18    27    14    14    14    14     3    27     4    30
326     7     7     7     7   205    16   249     5     7    14   200   200   172   270
END OF FILE
```

Figure A-3

The PREPROCESSOR Program

```
$LIST PROCESS/S
      IMPLICIT INTEGER (A-Z)
      DIMENSION DW(5),IW(42),C(120,9),DPP(120,5),IET(120,42),CC(120,8)
      DIMENSION CRT(120,8),CNT(120,8),WHICH(120),QRN(120),T(120)
      DIMENSION COST(120)
      REAL DW,X,IW,TIME,TMIN
C FIRST READ IN INFORMATION ON HARDWARE
      READ (5,10) NINST,NCOMP
C NINST IS NUMBER OF DISPLAY INSTRUCTIONS
C NCOMP IS NUMBER OF COMPUTER/CONTROLS BEING CONSIDERED
 10   FORMAT (I2/I3)
      DO 30 I=1,NCOMP
      READ (5,50,END=60) (C(I,J),J=1,9),(CRT(I,J),CNT(I,J),J=1,8),
     1(CC(I,J),J=1,8),(DPP(I,J),J=1,5),(IET(I,J),J=1,NINST)
C C IS A/N DESCRIPTION OF COMPUTER AND DISPLAY CONTROL
C CRT IS NUMBER OF CRT'S IN THIS CONFIGURATION
C CNT IS NUMBER OF DISPLAY CONTROLS IN THIS CONFIGURATION
C CC IS COST OF THIS CONFIGURATION
C DPR IS DISPLAYABLE PERCENTAGE OF TEST PATTERNS
C IET IS INSTRUCTION EXECUTION TIME
 50   FORMAT (9A4/16I1/8I8/5I10/14I5/14I5/14I5)
 30   CONTINUE
 60   CONTINUE
      NAMELIST /NAM1/ NCOMP,NINST
      WRITE (6,NAM1)
C NOW READ IN INFORMATION FOR A PARTICULAR APPLICATION
 5    CONTINUE
      READ (5,19) A
 19   FORMAT (A4)
      READ (5,20) REQCRT
 20   FORMAT (I1)
C REQCRT IS NUMBER OF DISPLAY CONSOLES NEEDED (R)
      READ (5,21) QR
 21   FORMAT (I4)
C QR IS THE QUANTITY OF INFORMATION WHICH MUST BE DISPLAYED
      READ (5,22) X
 22   FORMAT (F7.5)
C X IS FRACTION OF DISPLAY DIFFERENT FOR EACH CONSOLE
      READ (5,23) (DW(J),J=1,5)
 23   FORMAT (5F5.4)
C DW ARE WEIGHTS FOR 5 TEST PATTERNS
      READ (5,24) (IW(J),J=1,NINST)
 24   FORMAT (14F5.4/14F5.4/14F5.4)
C IW ARE WEIGHTS FOR THE NINST DISPLAY INSTRUCTIONS FOR THIS APPLICATIC
C
C WILL NOW FIND WHICH, IF ANY, DISPLAY CONTROL CAN DISPLAY ENOUGH
C INFORMATION
C
      DO 110 I=1,NCOMP
      Q=0
      DO 70 J=1,5
 70   Q=Q+DW(J)*DPP(I,J)
C Q IS AMOUNT DISPLAYABLE ON 1 CRT BY CONTROLLER
      MINJ=0
      MINCST=1000000
      DO 80 J=1,8
      IF(CRT(I,J).NE.REQCRT) GO TO 80
      IF(CRT(I,J).EQ.0) GO TO 90
      QTEMP=Q/(X*(CRT(I,J)/CNT(I,J)-1)+1.0)
```

Figure A-3 (continued)

The PREPROCESSOR Program

```
C QTEMP IS AMOUNT DISPLAYABLE ON ALL CRT'S
      IF(QTEMP.LT.QR) GO TO  80
C TRANSFER MEANS CAN'T DISPLAY ENOUGH
C NOW FIND IF THIS SYSTEM MINIMIZES COST FOR I
      IF(CC(I,J).GT.MINCST) GO TO 75
        MINJ=J
        MINCST=CC(I,J)
        MINC=QTEMP
 75   CONTINUE
 80   CONTINUE
 90   CONTINUE
      IF(MINJ.GT.0) GO TO 95
        COST(I)=0
        GO TO 110
 95   CONTINUE
      COST(I)=MINCST
      GRN(I)=MINC
      WHICH(I)=MINJ
C IF COST IS ZERO, MEANS FOR I THERE WAS NO SUITABLE CONFIGURATION
C WHICH(I) SAYS WHICH OF 8 POSSIBLE CONFIGURATIONS WAS SELECTED
      TIME=0.0
      DO 100 K=1,NINST
 100  TIME=TIME+IW(K)*IET(I,K)
      T(I)=TIME
C THIS IS NOW AVERAGE INSTRUCTION EXECUTION TIME
 110  CONTINUE
C NOW HAVE BEST CONFIGURATION FOR EACH I
      DO 140 I=1,NCOMP
      IF(COST(I).EQ.0) GO TO 140
      DO 130 J=1,NCOMP
      IF (COST(J).GT.COST(I).AND.T(J).GE.T(I)) COST(J)=0
C J COST MORE AND TOOK MORE EXECUTION TIME, SO OUT IT GOES
      IF(COST(J).EQ.0) GO TO 130
      IF(COST(I).EQ.COST(J).AND.GRN(I).GT.GRN(J).AND.T(I).LE.T(J))
     1 COST(J)=0
C J DISPLAYS LESS AND TAKES LONGER, SO IT GOES OUT
 130  CONTINUE
 140  CONTINUE
C ARE NOW READY TO PRINT OUT VALID CONFIGURATIONS, IN ASCENDING PRICE
      NUMOUT=0
 145  CUTI=0
      TMAX=0.0
      DO 150 I=1,NCOMP
      IF(COST(I).EQ.0) GO TO 150
      IF(T(I).LE.TMAX) GO TO 150
        TMAX=T(I)
        CUTI=I
 150  CONTINUE
      IF(CUTI.EQ.0) GO TO 200
      COST(CUTI)=COST(CUTI)/40
      JJ=WHICH(CUTI)
      WRITE (6,160) COST(CUTI),T(CUTI),(C(CUTI,J),J=1,9),CNT(CUTI,JJ),
     1 CPT(CUTI,JJ)
 160  FORMAT (I5,5X,I10,10X,9A4,I2,',',I1)
      NUMOUT=NUMOUT+1
      COST(CUTI)=0
      GO TO 145
 200  IF(NUMOUT.GE.16) GO TO 300
      TEMP=16-NUMOUT
      DO 210 I=1,TEMP
```

Figure A-3 (continued)

The PREPROCESSOR Program

```
     21C    WRITE (6,22C)
     220    FCRMAT ('99999     9999999999          CUMMY TC PAC OUTPUT')
     300    GC TO 5
            ENC
ENC CF FILE
```

Appendix B

THE OPTIMIZATION PROGRAMS, AND THEIR INPUT DATA

There are five principal program modules associated with the optimization. The first is a main program, which does little more than call upon two of the four other program modules. They are the input-output subroutines, and the optimization subroutine. The optimization routine in turn calls a subroutine which finds $T(R, \underline{X}, \underline{Z})$. If $R > 1$, the fifth program module, RQA, is called to evaluate T.

The main program is aware only of R and NCOST, the number of maximum costs (C'_{max}) for which the optimization is to be performed. These are obtained from the input subroutine, INMOD.

For each cost, an optimization is performed with $R=1$, to find the vector $\underline{A}$ which minimizes $TL(R, \underline{X}, \underline{Z})$ according to the algorithm of Figure 4-2. If R is in fact 1, $\underline{A}$ is optimum. If not, the optimization is called via OPT2(R) to find the true optimum, using the algorithm of Figure 4-3. OUTMOD is called to print results. The main program is listed in Figure B-1.

The input subroutine, INMOD, is part of the input-output module. It reads all the necessary input data, and calls initialization entries to several subroutines so that the input data is available to them. The output subroutine, OUTMOD, prints out a list of optimum display systems, and the corresponding vectors X (in hex), their costs, and response times.

The I/O module is listed in Fugure B-2. Figure B-3 is a typical input data set, wnich, by use of the program's READ statements and FORMATs, can quickly be correlated with the various program variables, the significance of which are thoroughly documented in the listing. In most cases, variables in the program bear the same name used in this report's text. Outputs from this program have been shown in Tables 6-14 to 6-25. Input is from unit 5, and output to unit 6.

The optimization subroutine implements the algorithms in Figures 4-1, 4-2, and 4-3. The set S' is stored in XARRAY. To conserve storage space, the set S is never actually formed; the comparisons used to reduce S to S' are made between each new X and the current S' before entering X into S'. The program is listed in Figure B-4. Output device 7 is used by this and other programs for diagnostic information.

The fourth program module, Figure B-5, evaluates $T(R, \underline{X}, \underline{Z})$ using RQA if required. It calculates the model's parameters from the input information. This includes the division of processing calculations described in Section 2. 7. 1. The array TIME holds the quantities t_i (Section 2. 4). If RQA is used, it ultimately holds T_i (Section 2. 7). TIME is used to calculate T in the internal function TIMEF. The RQA program module is documented in a University of Michigan Systems Engineering Laboratory internal memo by I. S. Uppal, dated 18 September 1968, titled, "IBM 360 Version of RQA-1."

A number of subroutines are called by the above programs, and are listed in Figure B-6. The first is STATE, which implements a mapping from the 11-dimensional state space of the display model to a uni-dimensional state index. It, in turn, uses BINCOF to find certain binary coefficients stored in an array. The array is initially calculated by the program FCTSET. DELT finds the change in the state index caused by a change in the 11-dimensional state space, by calling STATE. SETX converts from X to $\underline{X}$ (Section 4. 2), and XSTAR finds X* (Section 4. 2) from X. COST finds $C'(\underline{X})$ (Section 2. 6) for any vector $\underline{X}$. Finally, PACESS finds the file storage access probability $1 - e^{-\frac{x}{20}}$ (Section 2. 3).

Figure B-1

Main Program

```
$LIST MAINSOURCE
 C          MAIN PROGRAM TO PERFORM DISPLAY SYSTEM SYNTHESIS
 C
            IMPLICIT INTEGER (A-Z)
            RUN=1
 5          WRITE (6,10) RUN
 10         FORMAT ('1',T40,'OPTIMUM DISPLAY SYSTEM DESIGN PROGRAM,'
           1' RUN NUMBER ',I2////)
 C
 C          NOW GET INPUT DATA
 C
            R=INMOD(NCOST)
 C
 C          NCOST IS NUMBER OF SEPARATE OPTIMIZATIONS WHICH WILL BE DONE
 C
            I=1
 20         IF(I.GT.NCOST) GO TO 30
            T=OPT(1,I)
            IF(T.EQ.1) GO TO 25
 C          TRANSFER MEANS XARRAY IN OPT TOO SMALL. GO ON TO NEXT COST
            IF(R.EQ.1) GO TO 25
            T=OPT2(R)
 25         I=I+1
            GO TO 20
 C
 C          NOW PRINT OUT RESULTS
 C
 30         TRASH=OUTMOD(R)
            RUN=RUN+1
            GO TO 5
            END
END OF FILE
```

Figure B-2
Input-Output Program

```
$LIST IOMODULESOUR
C
C     FUNCTION CALLED BY "R=INMOD(NCOST)" TO PASS R AND NCOST EASILY
C
      FUNCTION INMOD(/NCOST/)
      IMPLICIT INTEGER (A-Z)
      REAL ARRIVE,URD,UMD,UMC,PAGCST,CPUCST,RTIME,TRIP,EPSI
      REAL PROB(100),BM(100),BR(100)
      DIMENSION CLINK(16),RLINK(16),NMLINK(16,10)
      DIMENSION CCORE(16),SZCORE(16),NMCORE(16,10)
      DIMENSION CBULK(16),SZBULK(16),NMBULK(16,10)
      DIMENSION CCOMP(16),TCOMP(16),NMCOMP(16,10)
      DIMENSION RCOST(50),RESULT(50),RTIME(50),X(4)
      DIMENSION XMASK(4),XSHIFT(4)
      DIMENSION NINST(100),C(100,10)
C
C     ARRIVE IS ARRIVAL RATE OF JOBS FROM EACH USER
C     URD IS SERVICE RATE OF REMOTE BULK STORAGE UNIT
C     UMD IS SERVICE RATE OF MAIN BULK STORAGE
C     UMC IS SERVICE RATE OF MAIN COMPUTER,
C     FOR AN ENTIRE INTERACTION CYCLE
C     PAGCST IS COST OF ONE PAGE-MONTH OF DISK STORAGE AT MAIN
C     COMPUTER
C     CPUCST IS COST OF CPU IF IT WERE USED ALL THE TIME THE
C     DISPLAY TERMINAL IS IN USE, ON A MONTHLY BASIS
C     CLINK,CCORE,CBULK,AND CCOMP ARE ALL COSTS OF VARIOUS HARDWARE
C     RLINK IS DATA LINK TRANSMISSION RATES, IN BITS PER SECOND
C     SZCORE IS CORE SIZE, IN  UNITS OF 12000 BITS
C     SZBULK IS SAME,BUT FOR BULK STORAGE AT REMOTE COMPUTER
C     TCOMP IS WEIGHTED EXECUTE TIME FOR REMOTE COMPUTER,
C     IN MICROSECONDS
C     NMXXXX IS 40 CHARACTER DESCRIPTION OF EQUIPMENT
C     RCOST(I) IS ARRAY OF MAXCOSTS TO BE USED IN OPTIMIZATION,
C     WHILE RESULT(I) AND RTIME(I) ARE RESULTS OF THAT OPTIMIZATION
C     AFTER OPTIMIZATION, COST(I) IS THE ACTUAL COST
C     NCOST IS NUMBER OF ENTRIES IN THE COST VECTOR
C     SECONDS, DONE AT REMOTE COMPUTER
C     NPPPC IS CORRESPONDING NUMBER, BUT FOR PRE- AND POST- PROCESSING
C     MODE
C     MSGLTH IS DATA LINK MESSAGE LENGTH, IN BITS
C     NXXXX IS THE NUMBER OF HARDWARE TYPES READ IN.  IT MUST
C     BE A POWER OF TWO.
C     R IS NUMBER OF TERMINALS  ATTACHED TO REMOTE COMPUTER
C     MAXPAG IS MAXIMUM NUMBER OF UNITS OF STORAGE EVER NEEDED
C     XMASK IS MASK FOR X1...X4, AFTER X SHIFTED RIGHT BY XSHIFT
C     THAT IS, AS FAR RIGHT AS X WILL GO WITHOUT LOOSING BITS OF
C     X(I)
C     NT IS NUMBER OF DIFFERENT INTERACTION TYPES,EACH OF WHICH IS
C     DESCRIBED BY A FOUR-TUPLE, CONSISTINT OF NINST,PROB,BM,BR.
C     NINST IS THE NUMBER OF INSTRUCTIONS REQUIRED
C     PROB IS THE PROBABILITY THAT THIS INTERACTION TYPE WILL OCCUR
C     BM IS THE NUMBER OF BRANCHES TO MAIN COMPUTER BULK STORAGE,
C     IF MAIN COMPUTER IS USED.
C     BR IS THE SAME FOR THE REMOTE COMPUTER
C
      READ (5,10) NLINK,NCORE,NBULK,NCOMP
   10 FORMAT (4I2)
      WRITE (6,11)
   11 FORMAT ('1',T28,'AN OPTIMUM DISPLAY SYSTEM  WILL BE CHOSEN'
```

Figure B-2 (continued)

Input-Output Program

```
     1,' FRCM AMONG THE FOLLOWINC SUBSYSTEMS')
      WRITE (6,22)
 22   FCRMAT ('0'/'0'/T10,'DATA LINK CCST, $',T35,'TRANSMISSION'
     1' RATE, BPS',T75,'DESCRIPTICN'/'0')
      I=1
 24   IF(I.GT.NLINK) GC TO 23
 15   REAC (5,20) CLINK(I),RLINK(I),(NMLINK(I,J),J=1,1C)
 20   FORMAT (I5,5X,I10,10X,10A4)
      WRITE (6,21) CLINK(I),RLINK(I),(NMLINK(I,J),J=1,10)
 21   FCRMAT (' ',T15,I5,T40,I10,T65,10A4)
      I=I+1
      GC TO 24
 23   WRITE (6,26)
 26   FCRMAT ('C'/'0'/T10,'CORE STCRAGE CCST, $',T35,
     1'STORAGE CAPACITY',T75,'DESCRIPTION')
      I=1
 27   IF(I.GT.NCORE) GO TC 29
      READ (5,20) CCORE(I),SZCORE(I),(NMCORE(I,J),J=1,10)
      WRITE (6,21) CCCRE(I),SZCCRE(I),(NMCORE(I,J),J=1,10)
      I=I+1
      GO TC 27
 29   WRITE (6,30)
 30   FCRMAT ('C'/'0'/T10,'BULK STCRAGE CCST, $',T35,
     1'STCRAGE CAPACITY',T75,'DESCRIPTION'/'0')
      I=1
 32   IF(I.GT.NBULK) GO TO 35
      REAC (5,20) CBULK(I),SZBULK(I),(NMBULK(I,J),J=1,10)
      WRITE (6,21) CBULK(I),SZBULK(I),(NMBULK(I,J),J=1,10)
      I=I+1
      GO TO 32
 35   WRITE (6,36)
 36   FORMAT ('0'/'C'/T10,'REMOTE CCMPUTER COST, $',T35,
     1'AVERAGE INSTRUCTION EXECUTICN TIME',T75,'DESCRIPTION')
      I=1
 40   IF(I.GT.NCOMP) GC TC 50
      READ (5,20) CCOMP(I),TCOMP(I),(NMCOMP(I,J),J=1,10)
      WRITE (6,21) CCCMP(I),TCOMP(I),(NMCOMP(I,J),J=1,10)
      I=I+1
      GC TO 40
 50   READ (5,55) (XSHIFT(I),I=1,4),(XMASK(I),I=1,4),ENDBIT
 55   FORMAT (4I2/4Z4/Z4)
      READ (5,100) NT
 100  FORMAT (I2)
      I=1
 9C   IF(I.GT.NT) GO TO 111
      READ (5,101) NINST(I),PRCB(I),BM(I),BR(I),(C(I,J),J=1,10)
 101  FCRMAT (I10,F10.3,2F10.0,T45,10A4)
      I=I+1
      GC TO 90
 111  CONTINUE
      WRITE (6,110)
      WRITE (6,120) (NINST(I),PRCB(I),BM(I),BR(I),(C(I,J),J=1,10),
     1I=1,NT)
 110  FCRMAT ('1',T28,'APPLICATICN CHARACTERISTICS'/////T10,'TASKS',
     1' PERFORMED IN RESPCNSE TC USER REQUESTS'//T5,'INSTRUCTIONS',
     2T2C,'PROBABILITY',T35,'MAIN CCMP ACCESSES',T60,'REMOTE COMPUTER',
     3' ACCESSES'//)
 12C  FCRMAT (' ',T3,I1C,T20,F7.4,T40,F10.0,T65,F10.0,T80,10A4)
      WRITE (6,56)
 56   FCRMAT ('1',T10,'CTHER CHARACTERISTICS ARE...'///)
```

Figure B-2 (continued)

Input-Output Program

```
  940              NAMELIST /NAM1/ NPPPC,MSGLTH,R,MAXPAG,SYSPAG
  950              NAMELIST /NAM2/ ARRIVE,URC,UMD,UMC,PAGCST,CPUCST,TRIP
  955              NAMELIST/NAM3/ MXITER,EPSI
  960              READ (5,NAM1)
  970              WRITE (6,NAM1)
  980              READ (5,NAM2)
  990              WRITE (6,NAM2)
  994              READ (5,NAM3)
  996              WRITE (6,NAM3)
 1000              READ (5,45) NCOST
 1002      45      FORMAT (I2)
 1005              READ (5,46) (RCOST(I),I=1,NCOST)
 1010      46      FORMAT (I5)
 1020      C       NOW CALL OTHER FUNCTIONS TO INITIALIZE THEM
 1030              TRASH=SETXI(XMASK,XSHIFT)
 1040              TRASH=OPTI(RCOST,RESULT,RTIME,CPUCST,ENDBIT)
 1050              TRASH=COSTI(CLINK,CCORE,CBULK,CCOMP,PAGCST,MAXPAG,SZCORE,SZBULK)
 1060              TRASH=RQARNI(RLINK,SZCORE,SZBULK,TCOMP,ARRIVE,URC,UMD,UMC
 1070             1,NPPPC,MSGLTH,MAXPAG,CPUCST,TRIP,MXITER,EPSI
 1075             2,NINST,PROB,BM,BR,C,NT,SYSPAG)
 1080              INMOD=R
 1090              RETURN
 1100              ENTRY OUTMOD(DUMMY)
 1110              WRITE (6,60)
 1120      60      FORMAT('1',T30,'RESULTS OF OPTIMIZATION'/'0',T5,
 1130             1'MONTHLY COST, $',T35,'AVERAGE RESPONSE TIME, SECONDS',
 1140             2T70,'X',T75,'EQUIPMENT COMPLEMENT',/' ')
 1150              I=1
 1160      65      IF(I.GT.NCOST) GO TO 80
 1170              TRASH=SETX(X,RESULT(I))
 1180              WRITE (6,70) RCOST(I),RTIME(I),RESULT(I),(NMLINK(X(1),J),
 1190             1 J=1,10),(NMCORE(X(2),J),J=1,10),(NMBULK(X(3),J),J=1,10),
 1200             2(NMCOMP(X(4),J),J=1,10)
 1210      70      FORMAT ('0',T10,I5,T40,F10.3,T68,Z4,(' ',T75,10A4))
 1220              I=I+1
 1230              GO TO 65
 1240      80      OUTMOD=0
 1250              RETURN
 1260              END
END OF FILE
```

Figure B-3

Typical Input Data

```
$LIST INPUTDATA(100)
  100     08080808   2-D DRAWING, R=1
  101     00068      0000000300          103 SERIES DATA LINK
  102     00098      0000001200          SWITCHED LINES - ASYNCHRONOUS
  103     00158      0000002000          SWITCHED LINES - SYNCHRONOUS
  104     00182      0000002400          PRIVATE VOICE GRADE LINES
  105     00650      0000040000          TELPAK A
  106     01000      0000075000          TELPAK B
  107     01350      0000125000          TELPAK C
  108     03050      0000500000          TELPAK D
  109     00200               4          1 CORE MODULE
  110     00450               8          2 CORE MODULES
  111     00650              12          3 CORE MODULES
  112     00850              16          4 CORE MODULES
  113     01050              20          5 CORE MODULES
  114     01250              24          6 CORE MODULES
  115     01450              28          7 CORE MODULES
  116     01650              32          8 CORE MODULES
  117     00000               0          NO BULK STORAGE
  118     00150              32          ONE MODULE, DEC DF32
  119     00225              64          DEC DF32, 1 DS32 EXPANDER
  120     00300              96          DEC DF32, 2 DS32 EXPANDERS
  121     00350             512          DEC RF08 +RS08 DISK UNIT
  122     99999                          DUMMY TO PAD OUTPUT
  123     99999                          DUMMY TO PAD OUTPUT
  124     99999                          DUMMY TO PAD OUTPUT
  125      1025             272          PDP-8 + DEC 340 DISPLAY             1,1
  126      1112             271          PDP-8 + 340 DISPLAY + EX. ARITH.    1,1
  127      1175             164          PDP-9 + DEC 340 DISPLAY             1,1
  128      1250             117          ADAGE AGT 10                        1,1
  129      3125             112          ADAGE AGT 50                        1,1
  130     99999      9999999999          DUMMY TO PAD OUTPUT
  131     99999      9999999999          DUMMY TO PAD OUTPUT
  132     99999      9999999999          DUMMY TO PAD OUTPUT
  133     00030609                 XSHIFT
  134     0007000700070007         XMASK
  135     0FFF
  136     19
  137                10     .001         0          1    TERMINATE EXECUTION
  138                50     .005         0          1    DELETE CURRENT PICTURE
  139               100     .005         1          2    GET MENU OF EXISTING PICTURES
  140               100     .005         2          3    PICK PICTURE FROM MENU
  141               100     .005         2          3    CREATE NEW PICTURE
  142               200     .005         1          2    NAME NEW PICTURE
  143               100     .005         4          6    SAVE CURRENT PICTURE
  144                50     .100         0          0    ENTER POINT
  145                60     .369         0          0    DRAW LINE FROM PREVIOUS POINT
  146               200     .050         0          1    ENTER TEXT ELEMENT
  147              2000     .025         0          1    ENTER ARC OF CIRCLE
  148               100     .075         0          0    DRAW NEW LINE
  149               100     .050         0          1    DELETE ELEMENT FROM PICTURE
  150               110     .030         0          1    TEMPORARILY ERASE ELEMENT
  151               500     .010         0          1    REDISPLAY ALL TEMP ERASED ELEMENTS
  152              3000     .025         0          1    SCALE PICTURE
  153               200     .025         0          1    TRANSLATE PICTURE
  154               200     .070         0          1    MOVE ELEMENT
  155                50     .150         0          0    IDENTIFY ELFMENT
  156      &NAM1
  157      NPPPC=100
```

Figure B-3 (continued)

Typical Input Data

```
158       MSGLTH=500
159       R=1
160       MAXPAG=200
161       SYSPAG=3
162       &END
163       &NAM2
164       ARRIVE=.2
165       URD=30.
166       UMD=13.
167       UMC=.25
168       PAGCST=.19
169       CPUCST=10000.
170       TRIP=1.C
171       &END
172       &NAM3
173       MXITER=100
174       EPSI=.005
175       &END
176      03
176.1    01335
176.2    01370
176.3    01440
```

Figure B-4
Optimization Program

```
$LIST OPTSOURCE
      FUNCTION OPTI(RCOST,RESULT,RTIME,CPUCST,ENDBIT)
      IMPLICIT INTEGER (A-Z)
      DIMENSION RCOST(50),RESULT(50),RTIME(50),S(11),XX(4),XXX(4)
      DIMENSION XARRAY(500),XARTIM(500)
      REAL RTIME,XARTIM,MINTIM
      REAL MINF,F,CPUCST,XTIME,RUNRCA
      LOGICAL SW,TEST,ENTRY
      OPTI=0
      RETURN
C     RCOST HAS SYSTEM COSTS IN IT
C     RESULT HAS SYSTEM CONFIGURATICN IN IT
C     RTIME HAS SYSTEM RESPONSE TIMES IN IT
C     XARRAY HAS FEASIBLE SCLUTICNS IN IT
C     XARTIM HAS LCWER BOUNC CN RESPCNSE TIME FOR FEASIBLE SOLUTIONS
      ENTRY OPT(R,CINDEX)
C     R IS NUMBER OF TERMINAL USERS
C     CINDEX IS INDEX INTC CCST VECTCR
C     INITIALIZE VARIABLES FIRST
      SSIZE=0
      SZXARR=0
      DATA ALLCNE/ZFFFFFFFF/
      XMIN=ALLCNE
      SPSIZE=0
      OPT=0
C
C     FIND ALL CCNFIGURATICNS WHICH ARE FEASIBLE (MEET CCST RESTRAINT)
C
      X=0
5     XTEMP=XSTAR(X)-1
      CX=COST(X)
      CXTEMP=COST(XTEMP)
C     DETERMINE WHERE CCST CF X LIES
      IF(CX.LE.RCOST(CINDEX)) GC TC 10
      X=XTEMP
      GO TO 60
C     TRANSFER MEANS CX TCC HIGH
C     IF STILL HERE, CX NOT TCC HIGH.  NOW SEE IF CXTEMP=C(X*-1) IS LOW
C     ENCUGH:  IF SO, WILL TRY TO INCLUDE XTEMP AS FEASIBLE POINT IN
C     XARRAY.  IF NOT, WILL TRY TC PUT X INTO XARRAY
10    IF(CXTEMP.GT.RCCST(CINDEX)) GC TC 20
      X=XTEMP
20    TRASH=SETX(XX,X)
      SSIZE=SSIZE+1
      SW=.TRUE.
C     SW SAYS X NOT YET PLACED INTC XARRAY
      SAVEI=0
C     WHEN SAVEI NOT 0, IT PCINTS TO AN AVAILABLE LCCATION IN XARRAY
C     BEGIN SEARCHING XARRAY FCR PLACE TO PUT I
      I=1
30    IF(I.GT.SZXARR) GC TC 52
      DATA EMPTY/ZF0000000/
      IF(XARRAY(I).EQ.EMPTY) GO TC 48
      TRASH=SETX(XXX,XARRAY(I))
      TEST=XX(1).GE.XXX(1).ANC.XX(2).GE.XXX(2).AND.XX(3).GE.XXX(3)
     1 .ANC.XX(4).GE.XXX(4)
C     TEST TRUE IF X SUBSUMES A ENTRY NOW IN XARRAY
      IF(.NCT.TEST) GC TO 50
      IF(.NCT.SW) GO TO 46
```

Figure B-4 (continued)

Optimization Program

```
560   C     TRANSFER IF X ALREADY PLACED IN XARRAY
570   C     HERE PUT X INTO XARRAY(I)
580         XARRAY(I)=X
590         SW=.FALSE.
600         GO TO 50
610   46    XARRAY(I)=EMPTY
620         GO TO 50
630   48    SAVEI=I
640   50    I=I+1
650         GO TO 30
660   C     CHECK TO MAKE SURE X GOT STORED
670   52    IF(SAVEI.GT.0.AND.SW) GO TO 53
675   C     TRANSFER MEANS CAN PUT X AT SAVEI
680         IF(.NOT.SW) GO TO 60
682   C     TRANSFER MEANS X STORED EARLIER
684   C     STAYING MEANS PUT X AT END OF XARRAY
690         SZXARR=SZXARR+1
700         IF(SZXARR.LE.500) GO TO 56
710         WRITE (6,55)
720   55    FORMAT (' ****TOO MANY POINTS IN XARRAY')
730         OPT=1
740         RETURN
742   53    XARRAY(SAVEI)=X
744         GO TO 60
750   56    XARRAY(SZXARR)=X
760   60    IF(X.GE.ENDBIT) GO TO 62
770         X=X+1
780         GO TO 5
790   62    IF(X.EQ.ENDBIT) GO TO 64
800   C     X GREATER THAN ENDBIT PATTERN: SOMETHING WRONG HERE
810         CALL ERROR
820         STOP 00011
830   64    WRITE (7,66) SZXARR,(XARRAY(I),I=1,SZXARR)
840   66    FORMAT ('0        XARRAY(1) TO XARRAY(',I3,')'//(8(1X,Z8)))
850         MINTIM=1000009.
860         GO TO 70
870   C
880   C     TRY ALL FEASIBLE POINTS TO FIND BEST ONE
890   C
900         ENTRY OPT2(R)
910         SPSIZE=0
920         OPT2=0
930   C     THIS SECOND ENTRY IS USED ONCE XMIN HAS BEEN OBTAINED USING
940   C     OPT(1,CINDEX)
945         XMIN=MINX
950         MINTIM=RUNRQA(R,XMIN,F)
960         SPSIZE=1
970         MINF=F
975         WRITE (7,68) XMIN,MINTIM
980   68    FORMAT (' AT ENTRY TO OPT2, XMIN=',Z5,' MINTIM=',F10.4)
990   70    I=1
1000  72    IF(I.GT.SZXARR) GO TO  76
1010        IF(XARRAY(I).EQ.EMPTY.OR.XARRAY(I).EQ.XMIN) GO TO 75
1020  C     EITHER IS DUMMY ENTRY OR HAS BEEN EVALUATED AS XMIN
1030        IF(R.GT.1) GO TO 73
1040        XTIME=RUNRQA(1,XARRAY(I),F)
1050        XARTIM(I)=XTIME
1060        GO TO 74
1070  73    IF(XARTIM(I).GE.MINTIM) GO TO 75
1080  C     TRANSFER MEANS NO HOPE OF GETTING A NEW MINIMUM HERE
```

Figure B-4 (continued)

Optimization **Program**

```
 1090            SPSIZE=SPSIZE+1
 1100            XTIME=RUNRQA(R,XARRAY(I),F)
 1110      74    IF(XTIME.GE.MINTIM) GO TO 75
 1120      C     HAVE FOUND SOMETHING BETTER
 1130            MINTIM=XTIME
 1135            MINF=F
 1140            MINX=XARRAY(I)
 1150      75    I=I+1
 1160            GO TO 72
 1170      76    RCOST(CINDEX)=COST(MINX)+F*CPUCST
 1180            RESULT(CINDEX)=MINX
 1190            RTIME(CINDEX)=MINTIM
 1200            WRITE (7,78) MINX,MINTIM,RCOST(CINDEX),R
 1210      78    FORMAT ('1**SUCCESSFUL OPTIMIZATION'/' X= ',Z5/'RESPONSE TIME = ',
 1220           1F10.4/'COST=',I5/'R=',I1)
 1230
 1240            IF(R.EQ.1) GO TO 82
 1250            WRITE (6,80) CINDEX,SSIZE,ENDBIT,SPSIZE
 1260       80   FORMAT (' OPTIMIZATION ',I3,', ',I3,' OF A POSSIBLE',I5,
 1270           1' CONFIGURATIONS WERE FEASIBLE.',I4,' WERE EXAMINED WITH RQA')
 1280            WRITE (6,83)
 1290      83    FORMAT (////'     SUBOPTIMUM SYSTEMS ARE'///)
 1300            DO 100 I=1,SZXARR
 1310            IF(XARRAY(I).EQ.EMPTY) GO TO 100
 1320            TRASH=SETX(XX,XARRAY(I))
 1330            WRITE (6,84) (XX(J),J=1,4),XARTIM(I)
 1340      84    FORMAT (' X1=',I2,' X2=',I2,' X3=',I2,' X4=',I2,'  T=',F10.5)
 1350     100    CONTINUE
 1360      82    RETURN
 1370            END
END OF FILE
```

Figure B-5

Program to Evaluate T with RQA

```
$LIST RQARUNSOURCE
C      THIS IS INITIALIZATION ENTRY
C
       FUNCTION RQARNI(RLINK,SZCORE,SZBULK,TCOMP,ARRIVE,URD,UMD,UMC,
      1NPPPC,MSGLTH,MAXPAG,CPUCST,TRIP,MXITER,EPSI,NINST,PROB,BM,BR,
      2C,NT,SYSPAG)
C
C      PROGRAM TO DETERMINE RESPONSE TIME FOR A DISPLAY SYSTEM
C      INITIALIZATION IS AT RQARNI,CALLED FROM INMOD
C      RQARUN IS THE MAIN ENTRY POINT,AND IS CALLED BY THE OPTIMIZATION
C      PROGRAM.
C
       IMPLICIT INTEGER (A-Z)
       DIMENSION NINST(100),C(100,10)
       DIMENSION XX(4),S(11),BEGIN(11),INCREM(11)
       DIMENSION RLINK(16),SZCORE(16),SZBULK(16),TCOMP(16),ARRAY(16,16)
       REAL PACESS,UMD,TRIP,EPSI
       REAL UCL,URCPPP,FRC,PCRC
       REAL ARRIVE,PN,URD,UMC,PD,CPUCST,PCORE,P1,P2,PEND
       REAL DENOM,TEMP,TIMEF,RUNRQA,ACC
       REAL PROB(100),BM(100),BR(100),TIMEM,TIMER,TRC,TMC,PMC,BMC,BRC
       REAL TRANS(8192),V(1400,2)
       REAL P(7,11),QUEUE(11),TIME(11)
       REAL F,T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11
       EQUIVALENCE (S(1),S1),(S(2),S2),(S(3),S3),(S(4),S4),
      1 (S(5),S5),(S(6),S6),(S(7),S7),(S(8),S8),(S(9),S9),
      2 (S(10),S10),(S(11),S11)
       EQUIVALENCE (TIME(1),T1),(TIME(2),T2),(TIME(3),T3),(TIME(4),T4)
      1,(TIME(5),T5),(TIME(6),T6),(TIME(7),T7),(TIME(8),T8)
      2,(TIME(9),T9),(TIME(10),T10),(TIME(11),T11)
       INTEGER*2 TABLE(3,8192)
       DUMFCT(ABCDEF)=ABCDEF
       TIMEF(UNIQUE)=(1.0-PN)/(1.0-PDRC)*(T1+PDRC*(PRD*T2+(1.0-PRD)*
      1(T4+2.0*T5)))+PN*(2.0*T6+2.0*T5+T8/(1.0-PD)
      2+PD/(1.0-PD)*T9)
       RUN=0
       INIT=0
       RQARNI=0
       TRASH=STATEI(S,R)
       TRASH=CELTI(S,T)
       TRASH=BIN(R,S,ARRAY)
       TRASH=FCTSET(ARRAY)
       RETURN
C
C      THIS IS ENTRY FOR EVALUATION OF RESPONSE TIME.  IF R=1, THE
C      CLOSED FORM SOLUTION FOR RESPONSE TIME IS USED
C      IF R>1, RQA IS USED.  IF R=100, THE DIVISION OF PROCESSING
C      IS PRINTED, AND RQA IS NOT USED.
C
       ENTRY RUNRQA(R,X,/F/)
       TRASH=SETX(XX,X)
C      FIND BRANCHING PROBABILITIES
       IF(SYSPAG.LE.SZCORE(XX(2))) GO TO 5
       RUNRQA=1000000.0
       F=1.0
       RETURN
5      CONTINUE
       PCORE=PACESS(SZCORE(XX(2))-SYSPAG)
       P1=PACESS(SZCORE(XX(2))+SZBULK(XX(3))-SYSPAG)-PCORE
```

Figure B-5 (continued)

Program to Evaluate T with RQA

```
480          P2=1.0-P1-PCORE
490          TIME(2)=1.0/URC
500          TIME(3)=MSGLTH/(RLINK(XX(1))*TRIP)
510          TIME(4)=1.0/UMD
520          TIME(5)=TIME(3)
530          TIME(6)=TCOMP(XX(4))*NPPPC*1E-6
540          TIME(7)=TIME(3)
550          TIME(9)=1.0/UMD
560          TIME(10)=TIME(3)
570          TIME(11)=TIME(6)
574          NAMELIST /NAM11/ T2,T3,T4,T6/NAM12/ T9,P1,P2,PCORE
575          NAMELIST /NAM13/ T1,T8,PN
580          TRC=0.0
590          TMC=0.0
600          PMC=0.0
610          BMC=0.0
620          BRC=0.0
625          NAMELIST /NAM10/TRC,TMC,BRC,BMC/NAM9/PMC
630          DO 37 K=1,NT
640          PEND=1.0/(1.0+BR(K))
650          DENOM=1.0-PCORE*(1.0-PEND)
660          IF(BR(K).GT.0) GO TO 10
670            PEND=1.0
680            DENOM=1.0
690     10   IF(ABS(P1+P2).LT..000001) GO TO 20
700            PRD=P1/(P1+P2)
710            PDRC=1.0-PEND/DENOM
720            GO TO 30
730     20   CONTINUE
740            PRD=0.5
750            PDRC=0.0
760     30   CONTINUE
764     C
765     C    DIVISION OF PROCESSING CALCULATED HERE
766     C
770          TIMEM=T6+T7+T10+T11+NINST(K)/(UMC*1000000.)+T9*BM(K)
780          TIMER=(NINST(K)*TCOMP(XX(4)))/1000000. + PDRC/(1.0-PDRC)*
790         1(PRD*T2 + (1.0-PRD)*(T3+T4+T5))
800          IF(TIMER-TIMEM) 35,35,36
810     C    GO TO 35 IF REMOTE COMPUTER IS FASTER
820     35   TRC=TRC+PROB(K)*(NINST(K)*TCOMP(XX(4)))
830          BRC=BRC+PROB(K)*BR(K)
840          IF(R.EQ.100) WRITE (6,40) (C(K,I),I=1,10)
850     40   FORMAT (' REMOTE COMPUTER;',10A4)
860          GO TO 37
870     36   TMC=TMC+NINST(K)*(PROB(K)/UMC)
880          PMC=PMC+PROB(K)
890          BMC=BMC+PROB(K)*BM(K)
900          IF(R.EQ.100) WRITE (6,41) (C(K,I),I=1,10)
910     41   FORMAT (' MAIN COMPUTER;',10A4)
920     37   CONTINUE
960          PN=PMC
965          IF(PN.EQ.0.000.OR.PN.EQ.1.000) GO TO 46
966     C    SKIP IF NORMALIZATION NOT NEEDED
970          TRC=TRC/(1.0-PMC)
980          TMC=TMC/PMC
990          BMC=BMC/PMC
1000         BRC=BRC/(1.0-PMC)
1005    46   CONTINUE
1010         PD=BMC/(1.0+BMC)
```

Figure B-5 (continued)

Program to Evaluate T with RQA

```
      NACESS=BRC
      WRITE (7,NAM9)
      WRITE (7,NAM10)
      IF(R.EQ.1.OR.R.FQ.100) WRITE (6,47) PN,(XX(KK),KK=1,4)
47    FORMAT (' PN= ',F10.9,'  X=',4I4)
C     NEW DC PROBABILITIES FOR THE FINAL ANALYSIS
      IF(R.NE.100) GO TO 45
      RQARLN=0
      RETURN
45    CONTINUE
      PEND=1.0/(1.0+NACESS)
      DENOM=1.0-PCORE*(1.0-PEND)
      IF(NACESS.GT.0)  GO TO 11
        PEND=1.0
        DENOM=1.0
11    IF(ABS(P1+P2).LT..000001) GO TO 21
        PRD=P1/(P1+P2)
        PCRC=1.0-PEND/DENOM
        GO TO 31
21    CONTINUE
        PRD=0.5
        PCRC=0.0
31    CONTINUE
      T1=(TRC*PEND)/(DENOM*1000000.0)
      T8=TMC*(1.0-PD)/1000000.0
      IF(PN.EQ.1.0) T1=1.0
      IF(PN.EQ.0.0) T8=1.0
      IF(R.GT.1) GO TO 50
      RUNRQA=TIMEF(DUMMY)
      F=PN*T8/(RUNRQA+1.0/ARRIVE)
      RETURN
C
C
C     BECAUSE R>1, MUST SET UP FOR AN RQA RUN
C
C
50    RUN=RUN+1
      NAMELIST /NAM2/TIME/NAM3/PCORE,P1,P2,DENOM
      WRITE (7,NAM2)
      WRITE (7,NAM3)
      CALL SETUP(TRANS,TABLE,V,8192,1400,1,RUN)
      UDL=1.0/TIME(3)
      URCPPP=1.0/TIME(6)
      WHERE=1
      DO 60 I=1,11
      BEGIN(I)=0
      S(I)=0
60    INCREM(I)=1
70    T=STATE(DUMMY)
      GO TO (100,200),WHERE
100   CONTINUE
C
C     SET UP QUADRUPLES
C
      STSUM=R-S1-S2-S3-S4-S5-S6-S7-S8-S9-S10-S11
      IF(STSUM.EQ.0) GO TO 110
        CALL QUAD(ARRIVE*STSUM*(1.0-PN),T,1)
        CALL QUAD(ARRIVE*STSUM*PN,T,DELT(0,6))
110   CONTINUE
      IF(S1.EQ.0) GO TO 120
```

Figure B-5 (continued)

Program to Evaluate T with RQA

```
           CALL QUAD((PRD*PDRC)/TIME(1),T,DELT(1,2))
           CALL QUAD((1.0-PDRC)/TIME(1),T,DELT(1,0))
         CALL QUAD((PDRC*(1.0-PRD))/TIME(1),T,DELT(1,3))
  120        CONTINUE
         IF(S2.GT.0) CALL QUAD(URC,T,DELT(2,1))
         IF(S3.GT.0.AND.S5.EQ.0) CALL QUAD(UCL,T,DELT(3,4))
         IF(S4.GT.0) CALL QUAD(UMD,T,DELT(4,5))
         IF(S5.GT.0) CALL QUAD(UDL,T,DELT(5,1))
         IF(S6.GT.0.AND.S1+S11.EQ.0) CALL QUAD(URCPPP,T,DELT(6,7))
         IF(S7.GT.0.AND.S3+S5+S10.EQ.0) CALL QUAD(UCL,T,DELT(7,8))
         IF(S8.EQ.0) GO TO 130
           CALL QUAD(PC/TIME(8),T,DELT(8,9))
           CALL QUAD((1.0-PC)/TIME(8),T,DELT(8,10))
  130    CONTINUE
         IF(S9.GT.0) CALL QUAD(UMC,T,DELT(9,8))
         IF(S10.GT.0.AND.S5+S3.EQ.0) CALL QUAD(UCL,T,DELT(10,11))
         IF(S11.GT.0.AND.S1.EQ.0) CALL QUAD(URCPPP,T,DELT(11,0))
         GO TO 300
  200    ACC=ACC+V(T,1)
  300    DO 400 I=1,11
         S(I)=S(I)+INCREM(I)
         IF(S1+S2+S3+S4+S5+S6+S7+S8+S9+S10+S11.LE.R) GO TO 70
         IF(S11.GT.R) GO TO 410
  400    S(I)=BEGIN(I)
  410    GO TO (1000,2000),WHERE
 1000    CONTINUE
C
C        HAVE NOW SET UP QUADRUPLES, AR READY TO USE RQA
C
         CALL DIAG(NTRANS)
         CALL SOLVE(NTRANS,MXITER,EPSI,MXSTAT,INIT)
         INIT=1
         DO 1010 I=1,11
 1010    QUEUE(I)=0.0
C
C        NOW COMPUTE MARGINAL DISTRIBUTIONS AND QUEUE LENGTHS
C
         WHERE=2
         USERNO=0
         QUENO=1
 2010    DO 2020 I=1,11
         BEGIN(I)=0
         S(I)=0
 2020    INCREM(I)=1
         BEGIN(QUENO)=USERNO
         S(QUENO)=USERNO
         INCREM(QUENO)=100
         ACC=0.0
         GO TO 70
C
C        RETURN BACK HERE AFTER COMPUTING A MARGINAL DISTRIBUTION IN ACC
C
 2000    P(USERNO+1,QUENO)=ACC
         QUEUE(QUENO)=QUEUE(QUENO)+ACC*USERNO
         ACC=0.0
         USERNO=USERNO+1
         IF(USERNO.LE.R) GO TO 2010
         USERNO=0
         QUENO=QUENO+1
         IF(QUENO.LE.11) GO TO 2010
```

Figure B-5 (continued)

Program to Evaluate T with RQA

```
C
C      DCNE.  NCW SET UP TIME(I) AS TOTAL THRCUGHPUT TIME
C      IN ORDER TO FIND RESPONSE TIME
C
       DO 2050 I=1,11
       IF(P(1,I).EQ.1.0) GO TC 205C
       TIME(I)=(TIME(I)*QUEUE(I))/(1.0-P(1,I))
2050   CONTINUE
       RUNRQA=TIMEF(DUMMY)
       F=PN*T8/(RUNRQA+1.0/ARRIVE)
3000   FORMAT (6(2X,F10.6)/)
       RETURN
       END
END OF FILE
```

Figure B-6

Subroutines

```
$LIST STATESOURCE+BINSOURCE+FACTSET+DELTSOURCE+SETXSOURCE+XSTARSOURCE+COSTSOURCE+PEXP/S
   4     C
   5             FUNCTION STATEI(S,/R/)
  10     C       STATE MAPPING FUNCTION
  11     C       INITIALIZED FROM CPT
  15             IMPLICIT INTEGER (A-Z)
  20             DIMENSION S(11)
  25             STATEI=0
  30             RETURN
  31     C
  32     C       MAIN ENTRY HERE
  33     C
  35             ENTRY STATE(DUMMY)
  45             SS=0
  50             II=2
  55     10      IF(II.GT.11) GO TO 40
  60             SSS=0
  65             JJ=1
  70     20      IF(JJ.GT.S(II)) GO TO 30
  75             SSS=SSS+BINCOF(JJ,II)
  85             JJ=JJ+1
  90             GO TO 20
  95     30      CONTINUE
 100             SS=SS+SSS
 110             II=II+1
 115             GO TO 10
 120     40      CONTINUE
 125             STATE=1+S(1)+SS
 130             RETURN
 135             END
   1     C       INITIALIZED FROM CPT
   2     C       FUNCTION TO COMPUTE A TERM IN SUMMATION FOR STATE INDEX
   3     C       CALCULATION.  BINOM(I+1,J+1) IS J THINGS DRAWN FROM I.
   4     C       INITIALIZATION ENTRY HERE
   5             FUNCTION BIN(/R/,S,BINOM)
  25             IMPLICIT INTEGER (A-Z)
  30             DIMENSION S(11),BINOM(16,16)
  41             BIN=0
  43             RETURN
  44     C       MAIN ENTRY HERE
  45             ENTRY BINCOF(JJ,II)
  50             SUM=0
  55             INDEX=1
  60     10      IF(INDEX.GT.(11-II)) GO TO 20
  65             SUM=SUM+S(12-INDEX)
  70             INDEX=INDEX+1
  75             GO TO 10
  80     20      CONTINUE
  85             TOP=R-JJ+II-SUM
  90             BINCOF=BINOM(TOP+1,II)
  95             RETURN
 100             END
   1     C       FUNCTION TO COMPUTE BINOMIAL COEFFICIENTS,PUT THEM
   2     C       INTO ARRAY(I+1,J+1)
   5     C
  10             FUNCTION FCTSET(ARRAY)
  15             IMPLICIT INTEGER (A-Z)
  20     C       FUNCTION TO COMPUTE BINOMIAL COEFFICIENTS,PUT THEM
  30     C       INTO ARRAY(I+1,J+1)
```

Figure B-6 (continued)

Subroutines

```
 40            DIMENSION ARRAY(16,16)
 41            DIMENSION FACT(8)
 43            DATA FACT/1,2,6,24,120,720,5040,40320/
 60            N=0
 70      1     M=0
 80      11    IF(M.EQ.N) GO TO 7
 90            T=N-M
110            IF(M.GT.N-M) GO TO 2
120            T=M
140      2     P=1
150            I=0
160      3     IF(I.GE.T) GO TO 4
170            P=P*(N-I)
180            I=I+1
190            GO TO 3
200      4     IF(T.EQ.0) T=1
201            ARRAY(N+1,M+1)=P/FACT(T)
202            ARRAY(M+1,N+1)=ARRAY(N+1,M+1)
205            ANSWER=ARRAY(N+1,M+1)
225            GO TO 8
230      7     ARRAY(N+1,M+1)=1
240      8     M=M+1
250            IF(M.LE.N) GO TO 11
260      9     N=N+1
270            IF(N.LE.15) GO TO 1
280            FCTSET=0
290            RETURN
300            END
  5      C
 10      C      SUBROUTINE TO FIND CHANGE IN STATE INDEX.
 15      C      PARA AND PARB SPECIFY WHICH STATE INDEX (S) TO DECREMENT
 20      C      AND INCREMENT, RESPECTIVELY.  IF ZERO, MEANS DO NEITHER.
 25      C      T IS THE CURRENT VALUE OF THE LINEAR STATE INDEX
 30      C      DELTI IS ENTERED ONLY ONCE, TO SET UP S ADDRESSING
 35      C
 40            FUNCTION DELTI(S,/T/)
 45            IMPLICIT INTEGER (A-Z)
 50            DIMENSION S(11)
 55            DELTI=0
 60            RETURN
 62      C
 63      C     MAIN ENTRY
 64      C
 65            ENTRY DELT(A,B)
 66            PARA=A
 67            PARB=B
 70            IF(PARA.EQ.0) GO TO 30
 75            IF(PARA.LE.11) GO TO 20
 80      10    WRITE (6,15)
 85      15    FORMAT(' ****PARA OR PARB TOO LARGE IN FUNCTION DELT')
 90            DELT=100000
 95            STOP 00001
105      20    S(PARA)=S(PARA)-1
110      30    IF(PARB.EQ.0) GO TO 40
115            IF(PARB.GT.11) GO TO 10
120            S(PARB)=S(PARB)+1
125      40    TEMP=STATE(TRASH)
130            IF(PARA.NE.0) S(PARA)=S(PARA)+1
135            IF(PARB.NE.0) S(PARB)=S(PARB)-1
140            DELT=TEMP-T
```

Figure B-6 (continued)

Subroutines

```
      RETURN
      END
C
C     X IS TO BE BROKEN DOWN INTO ITS COMPONENTS AND PLACED
C     XSHIFT SAYS HOW FAR RIGHT TO SHIFT X BEFORE MASKING IT
C     WITH XMASK
C     INITIALIZATION ENTRY HERE
      FUNCTION SETXI(XMASK,XSHIFT)
      IMPLICIT INTEGER (A-Z)
      DIMENSION XMASK(4),XSHIFT(4), XV(4)
      SETXI=0
      RETURN
C
C     MAIN ENTRY HERE
      ENTRY SETX(XV,X)
      I=1
   10 XV(I)=LAND(XMASK(I),SHFTR(X,XSHIFT(I)))+1
      I=I+1
      IF(I.LE.4) GO TO 10
      SETX=0
      RETURN
      END
C
C     SUBROUTINE TO FIND X*
      FUNCTION XSTAR(X)
      IMPLICIT INTEGER (A-Z)
      IF (X) 20,10,20
   10 XSTAR=1
      RETURN
   20 XSTAR=LOR(X,X-1)+1
      RETURN
      END
C     PROGRAM TO CALCULATE COSTS OF HARDWARE CONFIGURATIONS
C     INITIALIZED AT COSTI FROM INMOD
C     CALLED FROM OPT AT COST
C
C     INITIALIZATION ENTRY HERE
C
      FUNCTION COSTI(CLINK,CCORE,CBULK,CCOMP,PAGCST,MAXPAG,
     1SZCORE,SZBULK)
      IMPLICIT INTEGER (A-Z)
      REAL PAGCST,T
      DIMENSION CLINK(16),CCORE(16),CBULK(16),CCOMP(16)
      DIMENSION XX(4),SZCORE(16),SZBULK(16)
      COSTI=0
      RETURN
C
C     MAIN ENTRY HERE
C
      ENTRY COST(X)
      TRASH=SETX(XX,X)
      TEMP=MAXPAG-SZCORE(XX(2))-SZBULK(XX(3))
C     TEMP IS NUMBER OF FILES STORED AT MAIN COMPUTER
      T=TEMP*PAGCST
C     T IS COST IN DOLLARS FOR STORAGE AT MAIN COMPUTER
      TEMP=T
      COST=CLINK(XX(1))+CCORE(XX(2))+CBULK(XX(3))+CCOMP(XX(4))+TEMP
      RETURN
      END
C
```

Figure B-6 (continued)

Subroutines

```
      5     C     SUBROUTINE TO CALCULATE STORAGE ACCESS PROBABILITY
      6     C
     10           FUNCTION PACESS(INT)
     20           ARG = -INT/20.0
     30           PACESS=1.0-EXP(ARG)
     40           RETURN
     50           END
END OF FILE
```

Appendix C

DATA FROM DISPLAY APPLICATIONS USING IBM 2250 DISPLAY SYSTEM AND MICHIGAN TERMINAL SYSTEM

In this appendix data taken on different display applications will be presented. The purpose of this data is to compare the computational requirements of various display (and non-display) applications. The data has been collected for jobs running on the University of Michigan's IBM 360/67, using the Michigan Terminal System [57].

Imbedded within MTS's executive system is an efficient data collection system, the use and operation of which have been documented by T. B. Pinkerton in a University of Michigan CONCOMP Project Memorandum entitled "The MTS Data Collection Facility," dated June 1968. Basically, a data item is recorded (on tape) whenever an event pertaining to specified jobs occurs. Examples of the events are 1) the start of a CPU processing interval, 2) the end of a CPU processing interval, 3) page read in or page read out start and end, 4) acquiring or releasing virtual memory pages, and 5) I/O to terminals, printers, or tapes.

A program has been written to analyze the data for any job and produce a series of probability distributions and summary data for the following quantities.

1. User think time. This begins when the computer system is ready to accept new input from the user, and ends

when the input is completed with an end-of-line indication.

2. CPU time used during the think period.
3. Computer system response time. This begins at the completion of an input line, and ends when all output has been finished and the computer system is again ready to accept input.
4. Processing interval lengths during response periods. During a processing interval a job has exclusive use of the CPU, except for supervisor functions.
5. Number of processing intervals during a response period.
6. Number of characters in input lines.
7. Number of characters in output lines.

When the analysis program was originally conceived and implemented, some of its results were intended to be used as part of the application specification required by the display system model. A serious deficiency in the data collection facility became evident and frustrated this aim. The problem is that input-output information is gathered only at the MTS level. The display service routines for the IBM 2250 display console do not use the MTS I/O routines once a graphics application program has been loaded and started from the console until it has been terminated. Therefore to the data collection facility, running a graphics program appears as one long response time, despite the many user interactions generated with the light pen and function buttons. Because this was the case, gathering much in the way of useful statistics for display applications became impossible. All of use that can be garnished from the display applications statistics is CPU untilization during response periods, and also averaged over think and response periods. While this information will be shown to be useful, it is not what was anticipated. This information is in Table C-1.

The first application referred to in the table, Michigan's Own Mathematical System (MOMS) is used to manipulate and plot mathematical functions. All interaction is via the light pen. The second application, text editing, uses the light pen and keyboard to modify text displayed on the console.

The 2250 display console could also be used in MTS as a teletype, with a screen instead of printer and paper. Table C-2 shows pertinent data from this mode of operation. The first three applications consist of general program preparation, correction, and debugging. The last application, running the system program *TASKS, gives a listing of jobs active in MTS.

The data collection facility was also used to monitor all MTS users for about one hour. The statistics gathered from a random sampling of the monitored teletype terminals are given in Table C-3.

Finally, a small amount of data was collected from a remote display terminal using MTS. The system consists of a DEC 339

Application	Average Use of CPU by Program	Use of CPU by Program During Response Time	Elapsed Time, Seconds	CPU Time, Seconds	Average Processing Interval, Microseconds
Michigan's Own Mathematical System	4. 10%	4. 25%	1342	54. 8	7521
Michigan's Own Mathematical System	4. 90%	5. 30%	412	20. 0	6030
Michigan's Own Mathematical System	4. 95%	5. 50%	1184	58. 6	5795
Text Editing	2. 30%	2. 40%	5080	116. 0	5116
Text Editing	2. 80%	3. 30%	2380	66. 0	5038

Table C-1

Data Gathered for IBM 2250 Display Console Used for Graphics

Application	Average Use of CPU by Program	Use of CPU by Program During Response Time	Elapsed Time, Seconds	CPU Time, Seconds	Average Think Time, Seconds	Average Response Time, Seconds	Average Processing Interval Length, Microseconds
Program Preparation and Testing	1. 10%	4. 0%	1280	14. 0	42. 0	13. 70	6518
Program Preparation and Testing	0. 75%	9. 8%	2360	17. 7	20. 9	0. 98	4780
Program Preparation and Testing	4. 55%	12. 5%	2650	121. 0	22. 2	10. 00	7634
Executing *TASKS	0. 37%	11. 4%	247	0. 9	47. 0	1. 05	5092

Table C-2

Data Gathered for IBM 2250 Display Console Used as a Teletype

Average Use of CPU by Program	Average Use of CPU by Program During Response Time	Elapsed Time, Seconds	CPU Time, Seconds	Average Think Time, Seconds	Average Response Time, Seconds	Average Processing Interval Length, Microseconds
3.50%	4.8%	1525	53.0	0.85	1.84	6103
1.80%	3.1%	578	10.2	25.00	31.90	3316
1.20%	2.4%	297	3.7	16.10	15.80	5275
0.15%	4.4%	4760	7.0	194.00	4.60	5837
2.40%	4.5%	512	12.3	14.60	15.50	3998
4.50%	9.8%	779	34.7	42.60	35.20	5782
0.70%	1.2%	207	1.5	12.10	39.80	3947
1.10%	1.5%	83	0.9	14.00	28.60	4610
2.10%	2.9%	356	7.6	25.30	63.60	3462
1.10%	1.8%	1580	17.5	24.10	36.70	4454
0.20%	1.0%	3610	7.1	35.90	5.70	3438
0.90%	3.3%	2930	25.2	44.60	14.10	4807

Table C-3

Data Gathered for Random Teletype Users

Average Use of CPU by Display Terminal	7.30%
Average Use of CPU by Display Terminal During Response Time	13.30%
Elapsed Time, Seconds	800.00
CPU Time, Seconds	58.50
Average Display Terminal "Think Time," Seconds	2.50
Average Response Time, Seconds	2.96
Average Processing Interval Length, Microseconds	6007.00

Table C-4

Data Gathered for Remote Display Terminal

Application Class	Average Use of CPU	Average User Think Time	Average Response Time
Teletype	1.05%	22.8	24.40
2250 Display Used as Teletype	2.36%	23.0	6.40
2250 Display Used for Graphics	3.03%	*	*
Remote Display Used for Graphics	7.30%	*	2.96

* Not Applicable

Table C-5

Comparison of Statistics

with 16,384 words of core storage, connected to the main computer via a 2000 bits per second data link. The application is queueing network analysis, in which the user draws on the display a queueing network, and can then see various probability distributions pertaining to the network. All graphics work is done by the display terminal: the main computer merly solves the network for the required results. The data is in Table C-4. Only a very simple queueing network was analyzed. It can be expected that for more complicated and realistic models, the average CPU utilization of 7.3% would increase. Note here that "Think Time" does not refer to the user, but to the remote display terminal. Thus an average of 2.5 seconds after receiving a reply from the main computer, the remote computer sends a new request for service to the main computer.

Table C-5 compares and summarizes some of the more important statistics from the four modes of using MTS reported on in Table C-1 through C-4. Several points should be noted. First, using a display in lieu of a teletype reduces response time by a factor of 4, with think time remaining nearly constant. This is a consequence of the fast rate of output attainable with the display console. The input rate (and consequently think time) is unaffected because a keyboard is used in both cases. Second, the faster output rate results in higher CPU use by the terminal, because more computation is done in less time. This means that a display terminal user gets more done in unit time than does a teletype terminal user, and the display user should therefore have a shorter connect time. Third, the 2250 display used for graphics work takes more CPU processing than when it is used as a teletype. Last, the remote display terminal uses more CPU processing than any other application class. However this final point, being based on just one data set, must at best be regarded as tentative. Also, the 360/67 hardware configuration in use when the remote display data was taken differed from the hardware in use when the other data was taken.

All this data, then, gives very positive confirmation to the idea that displays in place of teletypes use the CPU more and increase response time, and that graphics-oriented work requires more CPU time than does teletype-oriented work.

Figures C-1 through C-9 show some of the probability distributions and the summary data found from the remote display terminal statistics. They are shown here because of their relevance to the work reported here. Superimposed on the original data points (indicated by asterisks) are various distribution functions. In the cases of Figures C-2 through C-7, the distribution is hyper-exponential, with means and standard deviations equal to those of the experimental data. The hyper-exponential distribution is used because it satisfies the requirements for queueing analysis, and can be matched to both the first and second order statistics of the data.

In Figure C-8, a negative exponential distribution was used, with mean equal to that of the data. Because the data's standard deviation is less than its mean, the hyper-exponential distribution could not be used. Also, Figure C-9 uses a geometric distribution, which satisfies the requirements for queueing analysis, in that it represents a random branch determining the number of CPU intervals during a response period.

The purpose of showing these superimposed distributions is to visually compare some experimental distributions with those which are theoretically required to perform a queueing analysis. It is because of the differences seen between the theoretical and experimental distributions that the work reported in Chapter III was undertaken. It was shown there that the distribution can be much less important than its mean.

```
MEAN                SIGMA          SAMPLES

  2557643.C          6649902.C       145     THINK TIME
    10393.4             3563.5       145     TOTAL CPU TIME USED DURING THINK PERIODS
  2962184.0         1CC45788.C       144     RESPONSE TIME
   39367C.6          1468383.C       144     CPU TIMES DURING RESPONSE PERIODS
     6CC7.1             999C.4      9437     CPU INTERVALS CURING RESFCNSE FERIODS
       65.5              258.C       144     NUMBER CF CPU INTERVALS CURING RESPONSE PERIODS
        5.8                1.8       145     INPUT MESSAGE LENGTHS
       19.3               29.7       144     CUTPUT MESSAGE LENGTHS

                    TIMES ARE IN MICROSECONDS

                    MESSAGE LENGTHS ARE IN CHARACTERS
```

Figure C-1

Summary Data for Remote Display Terminal

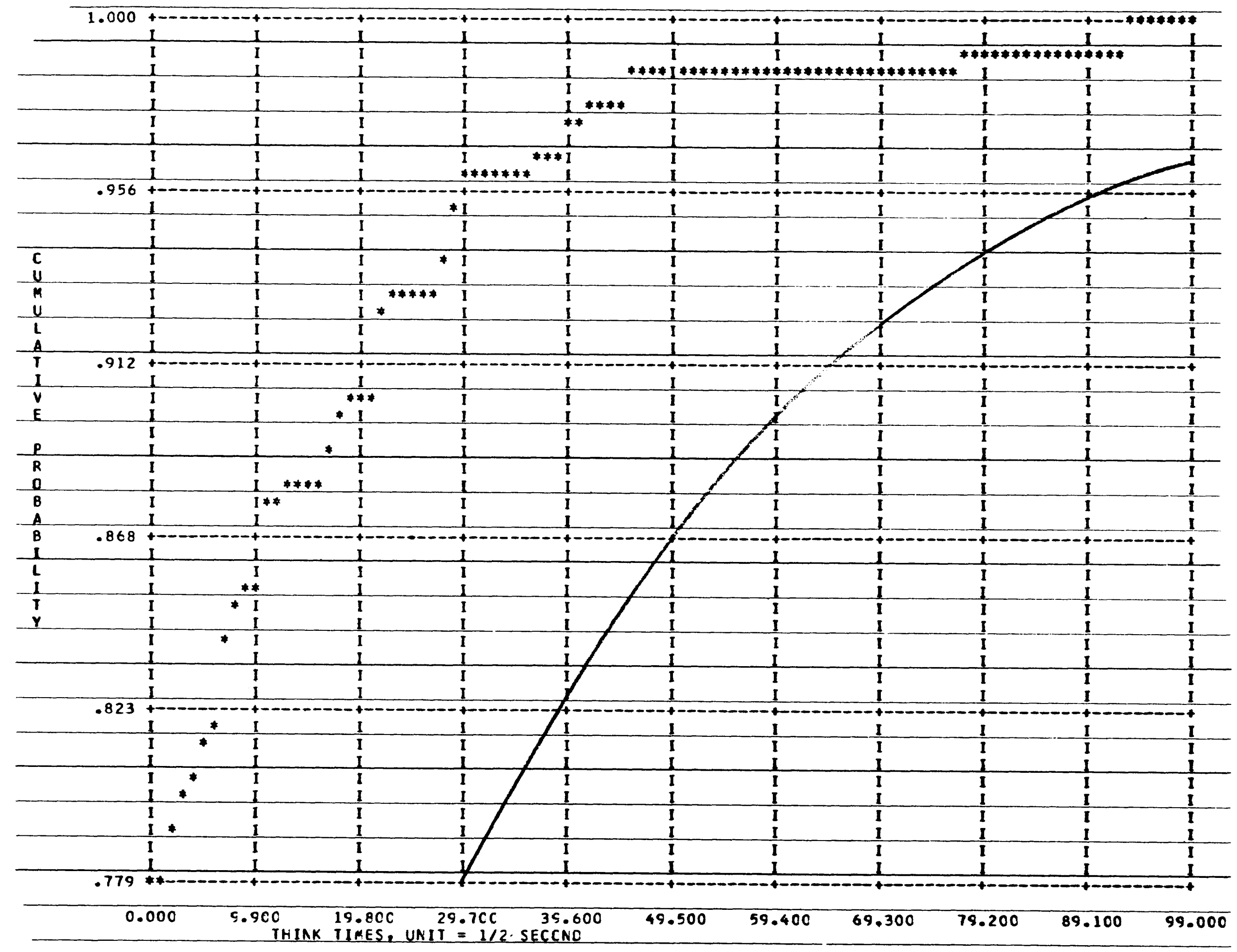


Figure C-2

Think Time Distribution

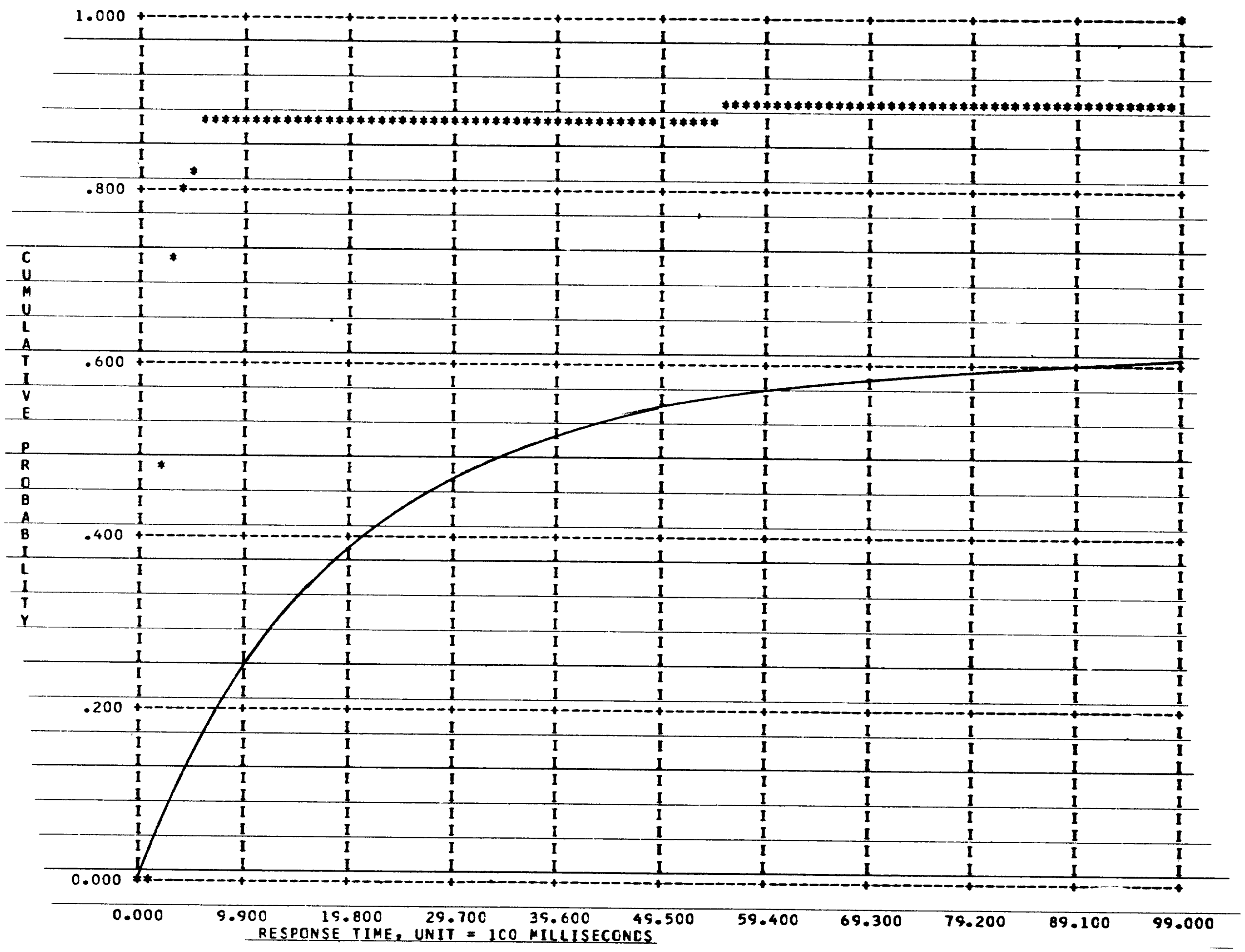


Figure C-3

Response Time Distribution

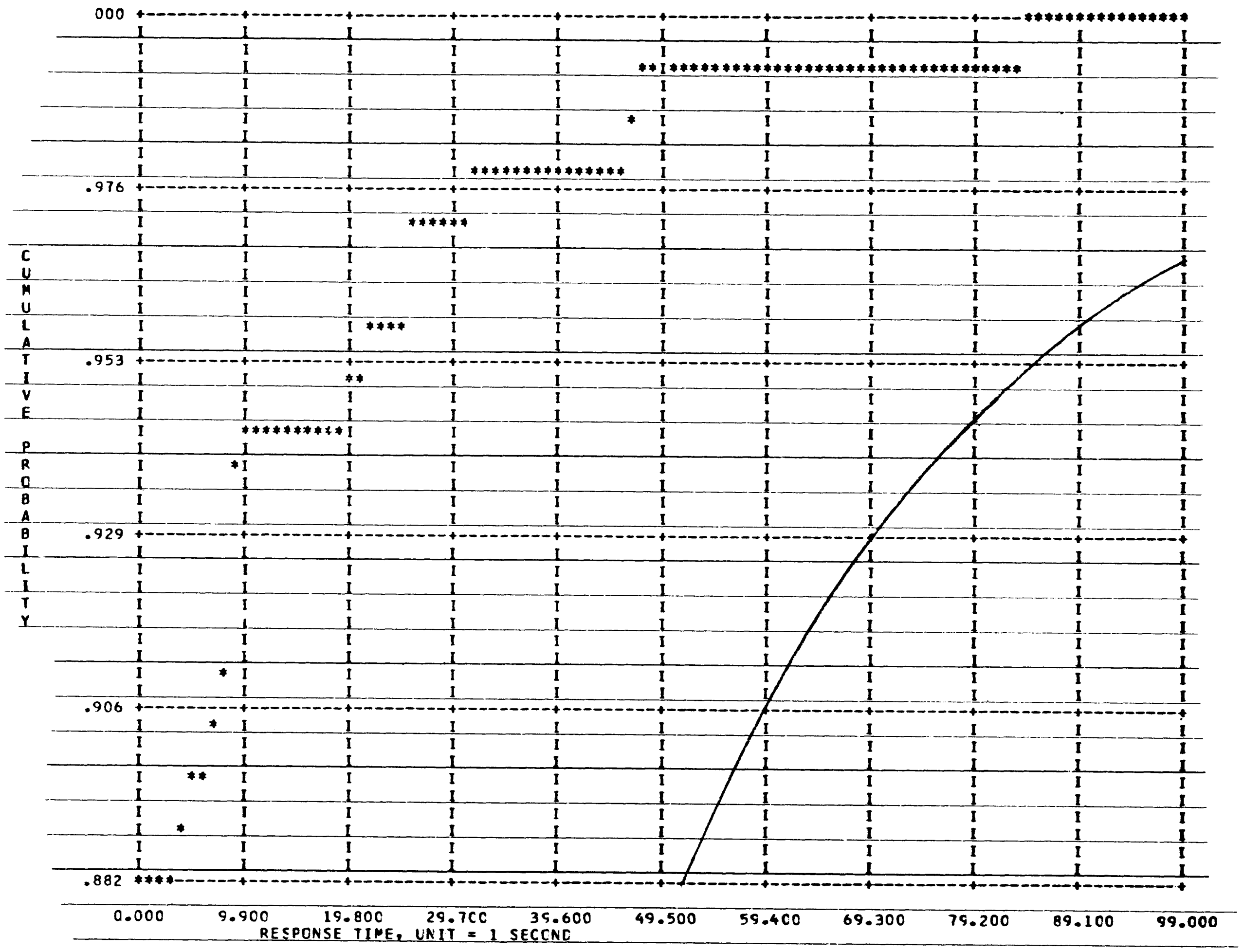


Figure C-4

Response Time Distribution

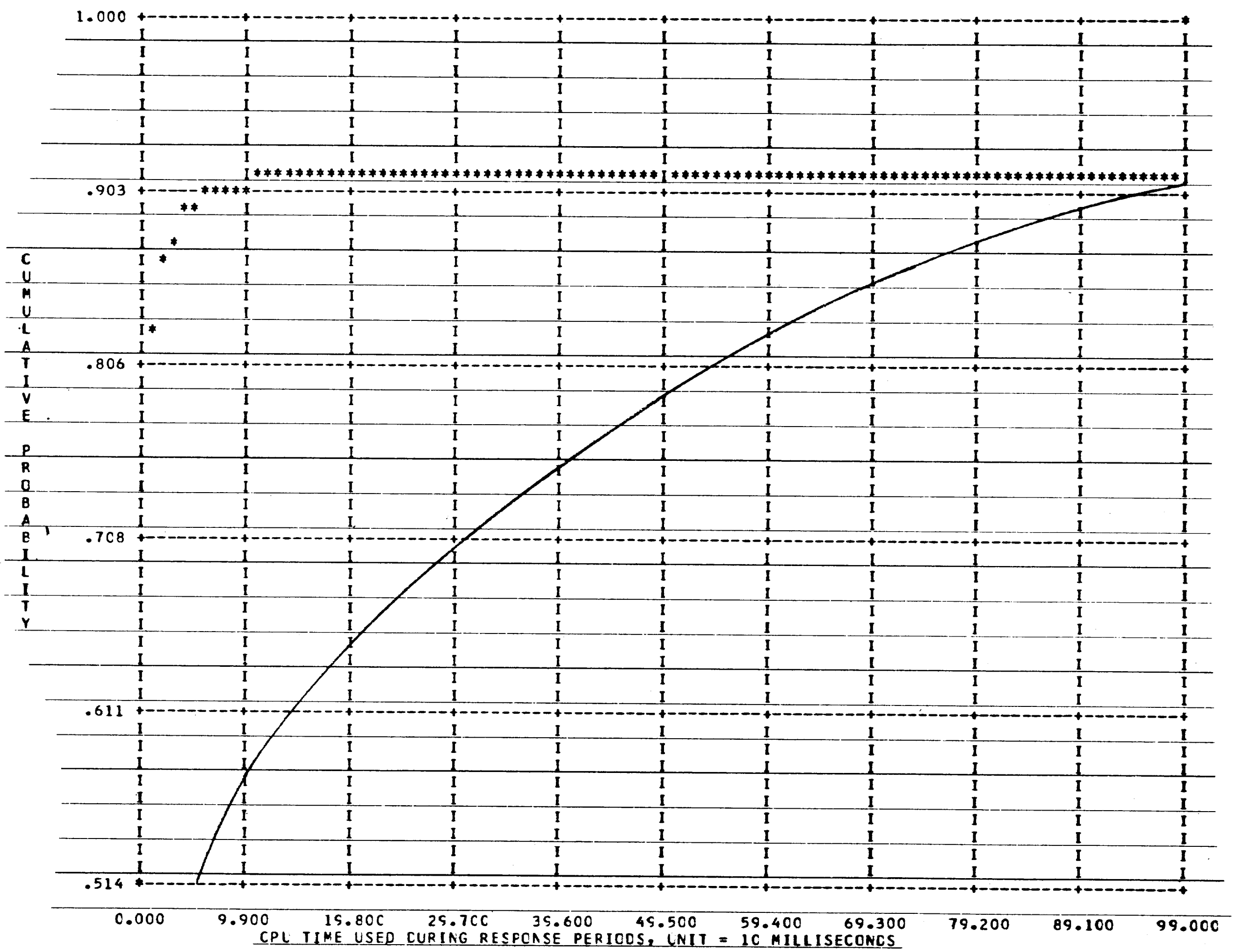


Figure C-5

Distribution of Total CPU Time per Response Period

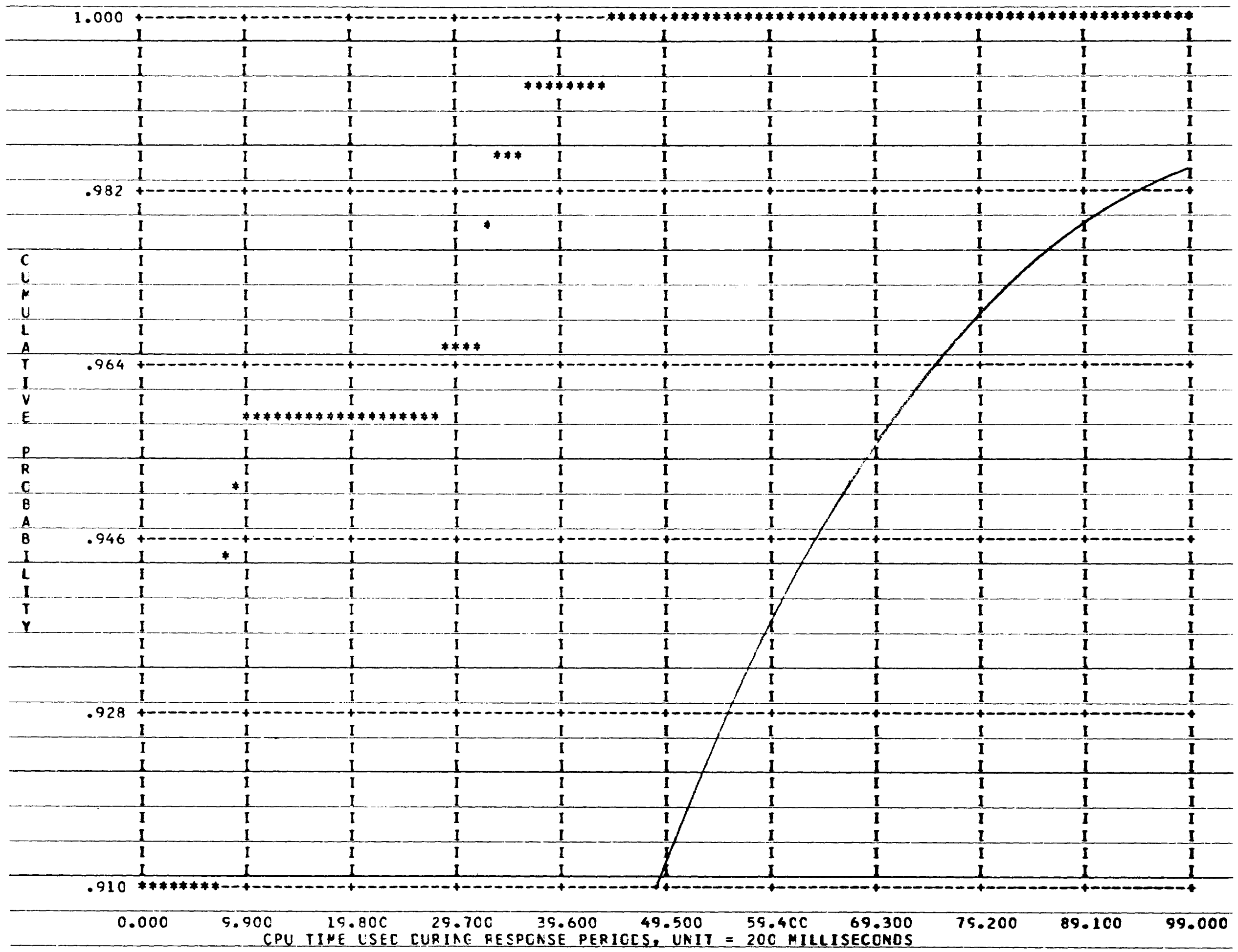


Figure C-6

Distribution of Total CPU Time per Response Period

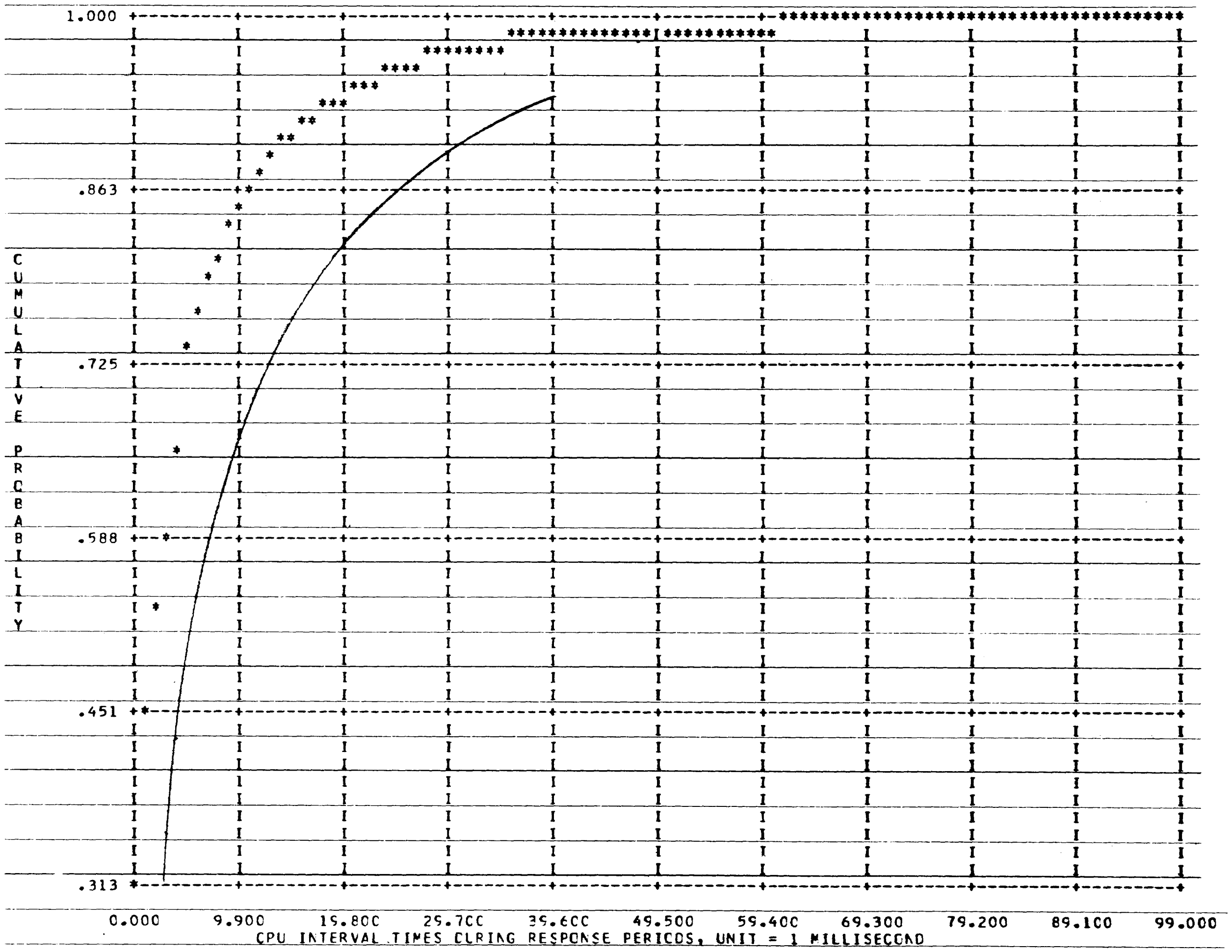


Figure C-7

Distribution of CPU Processing Interval Times During Response Periods

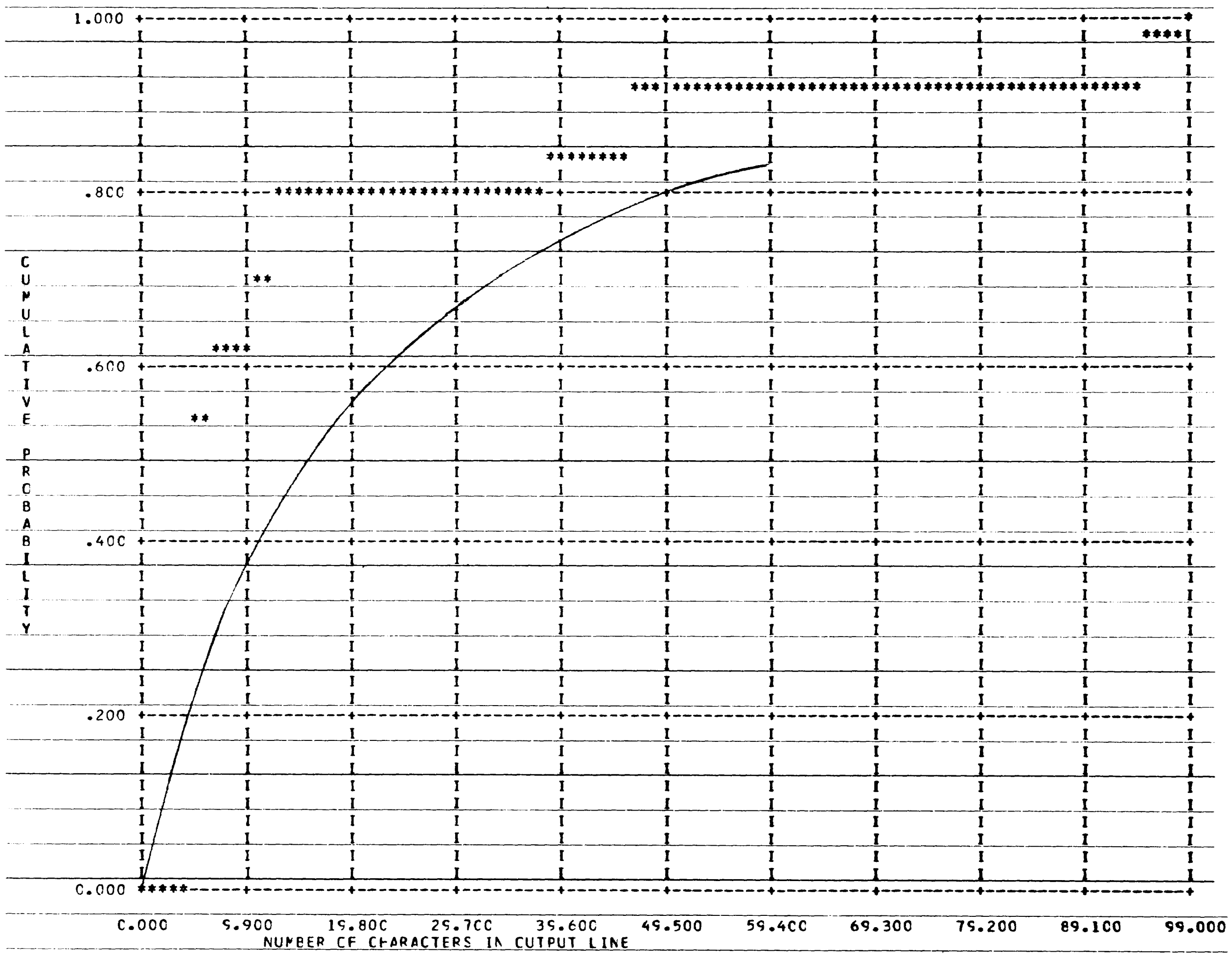


Figure C-8

Distribution of Output Line Lengths

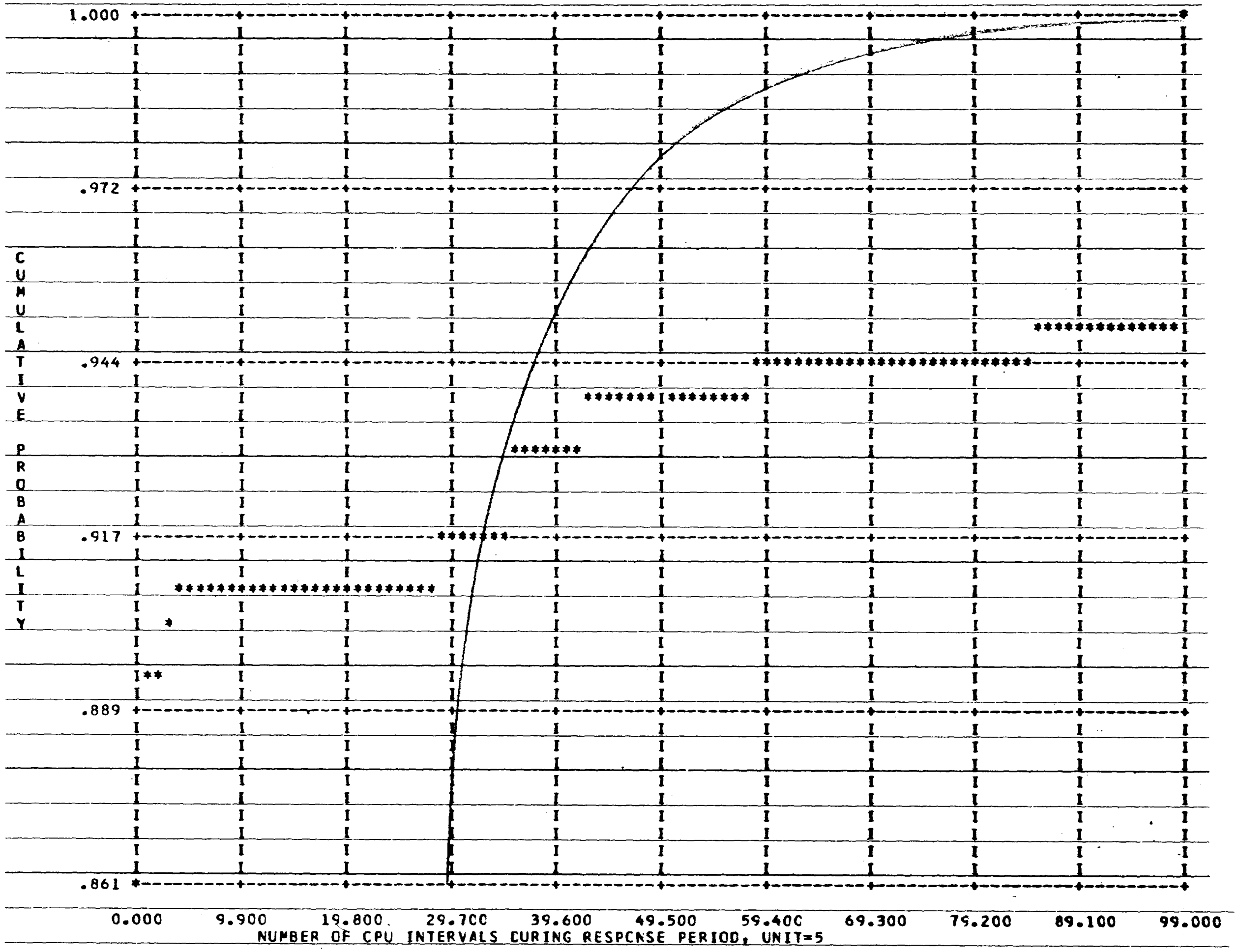


Figure C-9

Distribution of Number of CPU Intervals per Response Period