UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTING AND CONTROL

A Modelling Approach to the Evaluation of Computer System Performance

by

H. Gomaa

A thesis submitted for the degree of Doctor of Philosophy

November 1975

ABSTRACT

This thesis investigates some aspects of developing fast approximate models of computer system performance. Two different modelling techniques, regression and simulation modelling, have been applied and a method developed of combining their use within a multilevel hybrid modelling framework. The main objective of this thesis is to demonstrate the feasibility and value of this approach to the modelling and evaluation of computer system performance. The approach has been demonstrated by modelling a CDC 6000 computer system at three levels of detail. At each level, a self-contained model of the system has been developed. The Workload Model is a purely regression model of computer system performance. It was developed after a comprehensive performance analysis and analysis of residuals. The model expresses a batch job's elapsed time as a function of the job's resource demands and the load on the system. The model has been successfully validated. The Load Adjusting Model is a hybrid simulation/regression model in which a simulation framework is created which models job arrival and termination. Within this framework, regression techniques are used. The model has been successfully calibrated and validated. The Memory Management Model is developed from the Load Adjusting Model by a systematic expansion of detail. The Memory Management subsystem is simulated in considerable detail, whereas the rest of the system is modelled in much less detail. The model has been successfully calibrated and validated.

ACKNOWLEDGEMENTS

I am indebted to Professors D.J. Howarth and M.M. Lehman for their invaluable advice and encouragement. I am also indebted to Dr. H. Beilner for his invaluable advice and assistance and to W.J. Brooker for his assistance on statistical matters. I am also grateful to P.G. Jones and J.L. Thompson for implementing the Dayfile processing programs. Thanks are also due to Miss D. Pesko for typing this thesis so speedily and to Miss M. Cavill for her assistance. Last, but by no means least, I am very grateful to my wife Gill for her patience, understanding and encouragement.

CONTENTS

CHAPTER 1: INTRODUCTION

The measurement and evaluation of computer system performance is a two stage iterative process. In the first stage, the performance of a computer system processing a given workload, is measured. In the second stage, a performance evaluation is carried out, by analysing the data collected in the first stage, by modelling computer system performance, or by a combination of both. The insight gained by means of the evaluation, may well lead to a further iteration of the measurement and evaluation process.

Three major purposes for the evaluation of computer system performance are:

(a) selecting a computer system

  (i) where previously no computer was available

  (ii) to replace an existing system

(b) Performance measurement and analysis of an existing computer system to determine how effectively it is processing the workload applied to it. The objectives of this type of evaluation may be:

  (i) to determine the characteristics of the workload

  (ii) to determine throughput rates and response times under various load conditions

  (iii) to provide greater understanding of system performance

(iv) to detect bottlenecks or imbalances in system performance.

(v) to improve system performance as measured by some objective function such as turnaround time, response time or processor utilisation

(c) Performance and workload projection, that is predicting the performance of a given computer system:

(i) if the workload were to be changed

(ii) if changes were to be made to system parameters or scheduling algorithms

(iii) if modifications were to be made to the system configuration

Computer system modelling is a valuable tool in the evaluation of computer system performance. To gain the most advantage from a computer system model to be used for performance evaluation, it should be capable of modelling the system's performance in a fraction of the real world time. In such conditions, it is more economical to experiment with the model than with the system itself.

This thesis investigates some aspects of developing fast approximate models of computer system performance. Two different modelling techniques, regression and simulation modelling, have been applied and a method developed of combining their use within a multilevel hybrid modelling framework. The main objective of this thesis is to demonstrate the feasibility and value of this approach to the modelling and evaluation of computer system performance.

The approach has been demonstrated by modelling the Imperial College computer system at three levels of detail. At each level, a self-contained model of the system has been developed. The first level models workload performance using regression techniques. At the second level, simulation techniques are introduced and combined with the regression techniques. At the third level, more detail is introduced by simulating the memory management subsystem.

Chapter 2 surveys the field of performance measurement and evaluation of computer systems. Chapter 3 investigates the main aspects of computer system modelling and how they may be applied to the evaluation of computer system performance. Chapter 4 presents the main features of the computer system modelled, the Imperial College Control Data (CDC) 6000 system. Chapter 5 describes the CDC Dayfile, the only source of workload and performance data used in the evaluation. Chapter 6 describes the initial attempts at the regression modelling of the Imperial College system. These were largely unsatisfactory, but paved the way for a much more successful attempt at modelling the system with the development of the Workload Model, which is described in Chapter 7. An attempt to overcome some of the limitations of the purely regression Workload Model led to the development of a more detailed model, the Load Adjusting model, described in Chapter 8, in which simulation techniques are introduced and combined with the regression techniques. An attempt to overcome some of the limitations of the Load Adjusting model led to the development of the Memory Management model, described in Chapter 9, in which a more detailed simulation is carried out. Finally, Chapter 10 evaluates the work described in the thesis and makes proposals for future research.

CHAPTER 2: SURVEY OF PERFORMANCE MEASUREMENT AND EVALUATION OF COMPUTER SYSTEMS

2.1 Introduction

This chapter surveys the different tools and environments used in the measurement and evaluation of computer system performance.

In section 2.2, a survey of performance measuring (or monitoring) techniques is presented. The main features of the different types of performance monitors are outlined. In section 2.3, the need for different environments for performance evaluation is described. These environments are:

(i) The real computer system processing the real workload.

(ii) The real computer system processing a model of the workload.

(iii) A model of the computer system processing a model of the workload.

The means of creating these environments are presented and discussed. In sections 2.4, 2.5 and 2.6, the performance evaluation of each of these alternative environments is described, with examples from the evaluation of existing systems.

2.2 Performance Monitoring Techniques

2.2.1 Performance Monitors (G2, L6)

This section surveys tools which may be used for monitoring computer system performance. Different types of data collection techniques may be used for this purpose. These may be classified as hardware or software

monitors. Software monitors may either be sampling or event drivers monitors. Sampling monitors may either be external or internal to the operating system. Event driven monitors are usually internal monitors. More recently, some hybrid performance monitors have combined both hardware and software monitoring techniques (S6).

2.2.2 Hardware Monitors (B11, D6, N2)

A hardware monitor can sense and record the time and occurrence of system events and changes of state at the hardware level. The data collected may be cumulative (counts) or trace (time-stamped) and is often stored on some external medium such as magnetic tape. The data gathered usually provides measures of hardware resource activity, e.g. CPU and I/O channel activity, types of instructions executed, etc.

Although hardware monitors are capable of providing measures of total resource activity, it is frequently difficult to relate these measures to the executing workload in a multiprogramming system. In systems which distinguish by hardware between supervisor and user modes, the CPU time used in each of these two modes may be measured. Furthermore, in systems where a different hardware storage protection key is associated with each user partition (e.g. in OS/360 systems), the hardware monitor is capable of recording the CPU time used by each partition (B11). However, to correlate the CPU time used by a partition with the jobs executing in that partition, requires data on job commencement and termination times, which could only be obtained by software monitoring. To correlate I/O activity measured by a hardware monitor with the executing jobs would again need assistance from a software monitor.

2.2.3 Software Monitors

2.2.3.1 Monitoring Techniques

A sampling monitor is a software monitor which samples and records the state of the system at regular intervals. The data is often collected by reading the contents of system tables and queues, or by reading the contents of counters maintained by the system. The data gathered is usually blocked before being output onto an external storage medium, such as disc or magnetic tape. Examples of sampling monitors are EYE (S13) and KIK (L5) on CDC 6000 systems, CUE (H1) on IBM 360 systems, and SPASM (S3) on Burroughs 6700 systems.

A software event-driven monitor is an internal monitor which is implemented as a set of modifications to the operating system. As a result of this, whenever a specified event occurs, a transfer is made to a data gathering routine which may accumulate the data in counters, or provide a trace by time stamping a record of each event occurrence. The data is usually output in a similar manner to the sampling monitor. Examples of event driven monitors are GTF (I2) and DCF (P2) on IBM 360/370 systems, UTEX (S4) on CDC 6000 systems, and the software monitors used in the Multics (S1) and Honeywell H6000 (D3) systems.

In general, event-driven monitors are capable of gathering data of a more detailed nature than sampling monitors. An event-driven monitor is capable of recording a sequence of events where each event represents a change of state in the system. A sampling monitor can record sequences of states but not necessarily the changes in state.

More recently, some performance monitors have attempted to combine the advantages of both sampling and event driven techniques. V.M Monitor (C2, C3) is an internal monitor, imbedded within the IBM VM/370 operating system, which uses both techniques.

2.2.3.2 Data Collection

Data Collection may be carried out at different levels of detail and using different techniques depending on the type of monitor used and on the objectives of the subsequent evaluation.

(a) Level of Detail

(i) Workload Level

Data may be collected at the workload level, that is at job or job step level, e.g. job and job step execution times, resource utilisation, etc. This type of data is frequently collected by accounting systems, e.g. the Dayfile on CDC 6000 systems (G4) and System Management Facility on IBM OS/360 and OS/370 systems (I3).

(ii) System Event Level

Alternatively data may be collected at the system event level, e.g. each time the CPU is switched, a change in memory allocation occurs or an I/O operation is commenced or terminated. This method is associated with event driven monitors (S4), but may also be associated with a sampling monitor with a high sampling rate (L5).

(b) Methods of Data Collection

(i) Snapshot

This is the typical method used by a sampling monitor, which records the state of the system whenever it is activated (S13), e.g. which job is active, number of jobs waiting for CPU, number of jobs doing I/O, etc.

(ii) Cumulative

This method may be used by both sampling and event driven monitors. For example, counters of overall paging activity may be maintained by the operating system, and output at regular intervals by a sampling monitor (R1). Alternatively, a job's CPU time may be accumulated and output at job termination by an event driven monitor (G4).

(iii) Trace

Trace data is usually collected by an event driven monitor which records the type and time of occurrence of each event (S4). It may also however be collected by a sampling monitor which outputs a message when it notices that an event has occurred, which may of course be some time after the occurrence of the event (L5).

2.3 Environments for Performance Evaluation

2.3.1 The Need for Different Environments

A total computing system may be regarded as consisting of:

- a computer system, that is the computer hardware together with the operating system.

- the environment the system operates in, that is the workload processed by the system.

From the performance viewpoint, the output of such a system consists of performance measures of the computer system processing the workload applied to it (figure 2.1).

A user installation is usually most interested in analysing the normal production environment in which the computer system processes the user workload. However, in any complex computer installation, the workload varies with time of day and from month to month. This makes it difficult to quantify performance purely on the basis of measurement of the real workload.

For this reason, alternative environments for performance evaluation have been created, measured and analysed. These environments are created by modelling the user workload, the computer system, or both, as shown in figures 2.2 and 2.3 respectively. One method is to model the user workload and apply this to a real computer system (L6). By this means, experiments could be carried out in a controlled reproducible environment. An alternative is to create a model of the user workload in the form of a workload trace, which is then applied to a model of the computer system (C4).

2.3.2 Modelling the User Workload

A model of the user workload, to be applied to a computer system, should consist of a selection of jobs which are representative of the user workload. Experiments may then be carried out by either keeping the workload model constant, and changing operating system parameters or algorithms, or by keeping the system constant, and varying the workload model in a measurable fashion.

The two most frequently used methods for providing a reproducible environment are benchmarks programs and synthetic programs. A benchmark program is an existing user application program. A model of the user workload may be created by selecting a specific number of user jobs which are representative of the user workload (L6).

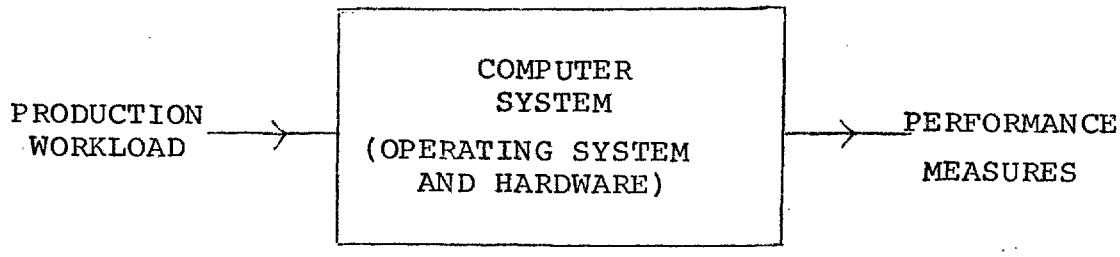


Figure 2.1: Performance Measurement of a Computer System Processing the Production Workload

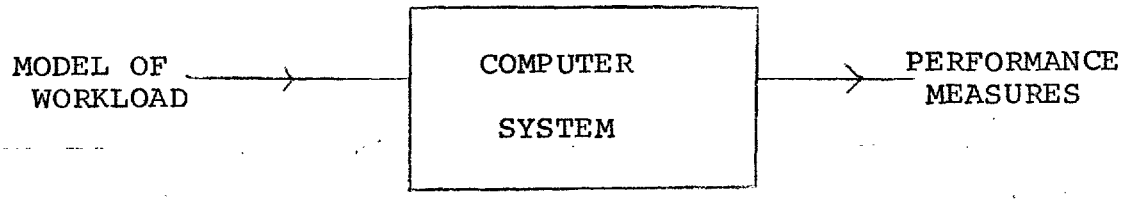


Figure 2.2: Real Computer System Processing Model of Workload

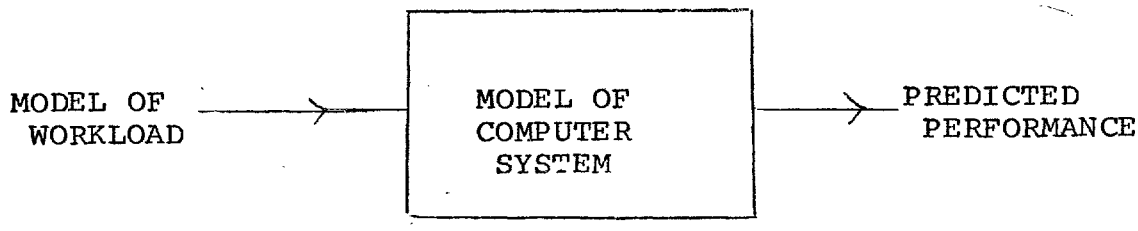


Figure 2.3: Model of Computer System Processing Model of Workload

A synthetic program is a program which has been specially created for the evaluation and does not serve any useful application apart from the evaluation (B13, K1). A synthetic program is fully parameterised in such a way that the same program may display widely different characteristics by simply changing the parameters. A typical synthetic program consists of a compute loop and an input/output loop. The ratio of how often each loop is executed determines whether the program is compute bound or input/output bound. In addition, a dummy array may be used to vary the size of the program (B13).

The degree to which results are meaningful depends on the degree to which the workload model is representative of the user workload (S11). This means that a thorough analysis of the real workload must first be carried out, so that the model may be calibrated against the real workload. It is likely that the calibration process is easier for a synthetic workload, because it is fully parameterised (O1). However, synthetic workloads are more difficult to construct for virtual storage systems.

The main advantages of using a model of the workload instead of the real workload are:

(i) It permits the creation of a controlled reproducible environment, where changes to the operating system may be evaluated having virtually eliminated the effect of a fluctuating workload.

(ii) Alternatively, the workload may be varied in a measurable fashion, and its effect on performance evaluated.

The main disadvantages of the controlled environment are:

(i) Calibrating the workload model against the real workload is likely to be a complex process (01). If the model is not calibrated, results may be unreliable.

(ii) Experiments using a controlled environment are time consuming and require a dedicated machine. Thus they can prove to be very expensive.

(iii) A workload is usually evolving, so the workload model would need regular updating.

(iv) The model of the workload is not the real workload. Some experimental aspects can lead to distorted measurements. For example, the running up and down time which occurs when executing a synthetic workload leads to 'edge effects'. These can be reduced but not eliminated by making the experimental run time long compared with the running up and down time.

2.3.3 Modelling the Computer System and Environment

2.3.3.1 Introduction

To model a total computer system for performance evaluation purposes, it is necessary to model both the computer system and its workload. A model of a computer system is an abstraction of the computer system's real world behaviour. Techniques for modelling computer systems include simulation, analytical and empirical techniques.

2.3.3.2 Analytical Models

Analytical models are mathematical models of computer systems and are often based on queuing theory. Queuing models of total computer systems usually involve a number of simplifying assumptions to make the model more amenable to mathematical analysis (G7, P1). These assumptions tend to reduce the validity of analytical models in computer system performance evaluation. Analytical models have probably been of most use in modelling subsystems, e.g. CPU scheduling (K4, M1), memory management (D2, B9), and I/O scheduling (T1, D3). Most of the work done, however, has not directly related the predictions of the analytical submodels to system performance (K2).

With the current state of the art, queuing models alone are therefore not sufficient for the evaluation of computer systems, but they can help in the understanding of systems (G7). Other methods must be used in conjunction with them.

2.3.3.3 Regression Models

Regression analysis is a method used to determine statistical relationships between two or more variables. A regression model is a functional relationship which relates an output (dependent) variable to a set of input (independent) variables. The functional relationship could be linear in its coefficients, such as:

$$Y = a_o + \sum_{i=1}^{k} a_i X_i$$

where a_i, i=0, 1, ... k are the regression coefficients, whose values may be determined by means of least squares fitting techniques (D1, D5).

Regression analysis has been used in performance evaluation to model workload performance, to estimate system overhead (B1) and to estimate the effect of a system change on performance in the normal production environment (W1, W4).

2.3.3.4 Simulation Models

A computer system simulation model models the system's real world behaviour by means of an algorithmic abstraction of the system, reflecting system structure and logical procedure.

Attempts have been made to provide one simulation program which models many different computer systems, but this necessitates the construction of such a gross model, that the results are usually only of general interest. Attempts to simulate a range of computers with basically the same hardware and operating system (e.g. IBM 360 and OS/360) have been more successful (S5). Many simulation models have been developed of individual computer systems (L3, N1, N3, W2).

2.3.3.5 Advantages and Disadvantages of Computer System Modelling

To gain the greatest advantage from a computer system model to be used for performance evaluation, it should model the system's performance in a fraction of the real world time. In such conditions, it is more economical to experiment with the model than with the system itself. Thus it becomes possible to experiment with a wide range of situations, which would not be practical in a real user environment (S7). Experimenting with the model has the additional important advantage, in that it may be used to predict the effect of configuration, system or workload changes.

The main disadvantage of modelling is its relatively high cost in the form of the time and man-power required to design, implement, calibrate and validate a model of the computer system. Another disadvantage is that from the point of view of experimenting with the model the most desirable area for experimentation may be an area for which it is difficult to determine the model's validity. The problems in calibrating and validating simulation models have been comprehensively described by Beilner (B4).

2.4 Performance Analysis of the Production Environment

2.4.1 Introduction

This section describes the performance evaluation of the production environment, that is the real computer system processing the real workload. The evaluation is presented according to the source of data available: collected by hardware monitors, from accounting data, by sampling monitors and by event driven monitors. Some evaluations have been carried out using more than one source of data, e.g. from accounting data and a sampling monitor (G5, S10).

2.4.2 Performance Analysis using Hardware Monitors

As described in section 2, a hardware monitor may be used for monitoring hardware resource utilisation over prolonged periods of time. From the data collected by a hardware monitor a system profile may be constructed. A typical example of this is shown in figure 2.4, which displays CPU and I/O channel utilisation over a complete session. From this type of data, it is possible to determine hardware resource bottlenecks or imbalances. For example, if CPU activity was much higher than I/O activity, this might suggest that a more powerful CPU

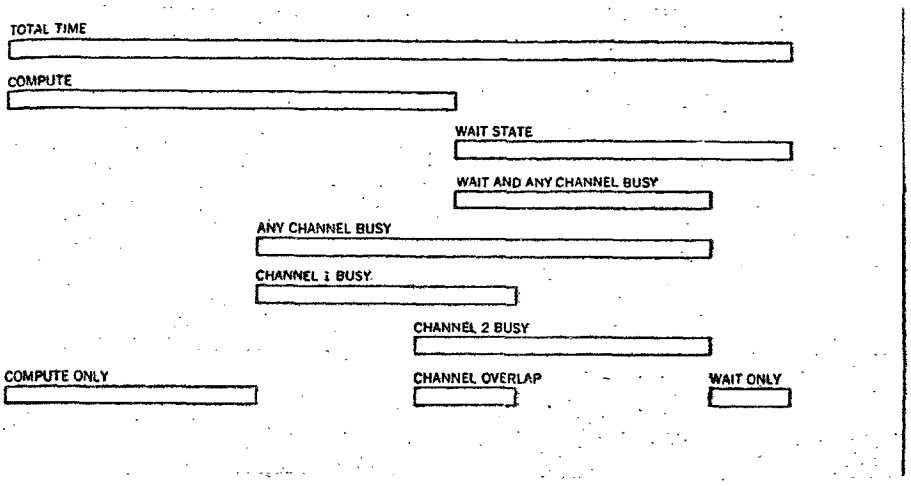


Figure 2.4: System Profile - Derived by Analysis of Hardware Monitor Measurements (B11)

was required. If I/O activity dominated, then possibly more disc space and another channel were required. If I/O activity is much higher on one channel than the others then the possibility of redistributing the files should be investigated (B12).

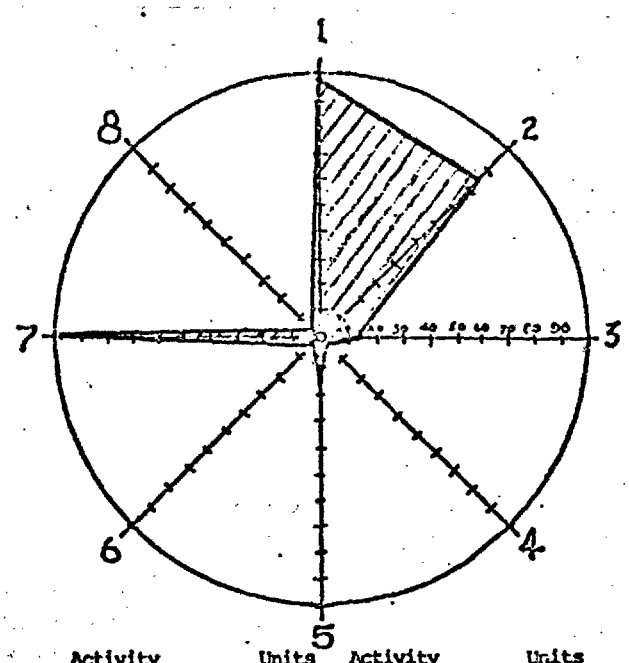
Another way of displaying performance data relating to resource activity is by means of a radial plot, attributed to Kiviat (M2). Its purpose is to display a large number of system variables in a geometric pattern which portrays different shapes characteristic of different loadings on the system. An example is given in figure 2.5 using an eight variable graph. The first diagram shows a CPU bound situation, while the second diagram shows an I/O bound situation.

2.4.3 Performance Analysis using Accounting Data

To relate hardware resource utilisation to the executing workload needs some form of software monitoring. The most common form of software monitoring found on computer systems is that used for accounting purposes, such as the Dayfile on CDC 6000 Systems (G4) and the System Management Facility (SMF) on IBM OS Systems (I3). These are basically internal monitors which gather data (part cumulative and part trace) at the macro level. Analysis of this data can provide:

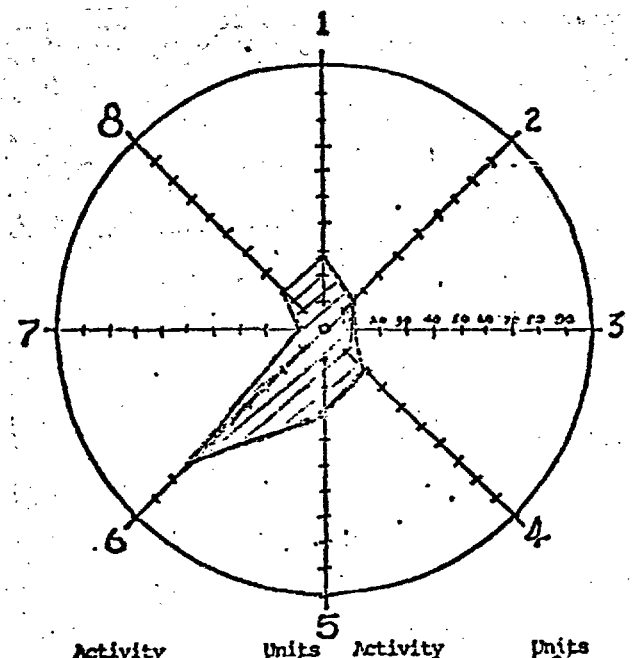
(i) a detailed workload profile of the system. For example, the distribution of jobs or job steps by their resource (CPU, Memory, I/O) utilisation or compiler usage.

(ii) System throughput rates. For example the mean and standard deviations of elapsed times for different classes of jobs at different times of the day/month/year.



Activity	Units %	Activity	Units %
1 CPU Active	98	2 CPU Only	84
3 CPU/Channel O'lap	14	4 Channel Only	2
5 Any Channel Busy	16	6 CPU Wait	2
7 Problem State	97	8 Supervisor State	1

CPU bound situation



Activity	Units %	Activity	Units %
1 CPU Active	28	2 CPU Only	17
3 CPU/Channel O'lap	11	4 Channel Only	21
5 Any Channel Busy	32	6 CPU Wait	71
7 Problem State	8	8 Supervisor State	20

I/O bound situation

Figure 2.5: Radial Plots - Derived by Analysis of Hardware Monitor Measurements (M2)

(iii) Information on peak demands and loads on the system. For example, number of terminal users logged in, length of job input and output queues, CPU utilisation and its distribution between, for example, the batch and interactive workloads.

This information can be used to accumulate long term workload statistics, enabling the installation management to determine long-term trends in the workload profile,throughput rates and peak demands. Letts (L4), Stanley (S12),Watson (W4) and Landau (L1) have all described how valuable workload and performance statistics were derived from this type of analysis.

2.4.4 Performance Analysis using Sampling Monitors

Sampling monitors usually collect data at regular intervals determined by some sampling interval. The data may be either snapshot data or from counters which have been accumulated by the operating system during the elapsed interval. One of the earliest sampling monitors was developed by Stevens (S13). A peripheral processor on a CDC 6600 was used to monitor the system. The monitoring program EYE had two loops, a 4 millisecond loop which was mainly used to monitor CPU activity and a slower loop which took a more general snapshot of the system at 15 second intervals, (see figure 2.6). Such a monitor can provide a valuable indication of the loads on the various resources and can identify bottlenecks occurring during a monitored session. However it is less likely to provide sufficient data to enable the cause of the bottleneck to be determined.

OPERATING CHARACTERISTICS FROM 15.56.41 TO 17.15.13, 08/22/67.

TOTAL COUNT= 1145868 (ONE COUNT EVERY 4.096 MILLISECONDS)
CP IDLE TIME= 391989 (34.081 PERCENT)
NON-COMPILE= 677389 (59.910 PERCENT)
COMPILE TIME= 80591 (7.009 PERCENT)
JOBS STARTED= 75

2.61 CHANNELS (INCL CP) ACTIVE

NUMBER OF ACTIVE PPUS	CONTROL POINTS: USER	OPERATOR	EMPTY	RECALL	COMPILING	ACTIVE CHANNELS: 0 1 2 3 4 5 6 7	0 1 2 3	TAPES ASSIGNED	DISK TRACKS ASSIGNED	MEMORY IN USE
2	3	1	3	1	1	A	A	0	190	331300
1	3	1	3	0	1		A	0	190	331300
2	3	1	3	1	2	A	A	0	191	331300
2	3	1	3	1	2	A	A	0	190	331300
1	3	1	3	1	2	A	A	0	190	331300
2	3	1	3	1	2	A	A	1	195	331300
2	3	1	3	1	2		A	1	195	331300
0	3	1	3	0	2		A	1	197	331300
1	3	1	3	0	2		A	1	197	331300
1	3	1	3	0	2		A	1	197	331300
1	3	1	3	0	2		A	1	198	331300
1	3	1	3	0	2		A	1	198	331300
1	3	1	3	0	2		A	1	198	331300
2	3	1	3	0	2	A	A	1	199	331300
2	3	1	3	1	2		A	1	198	331300
2	3	1	3	1	2		A	1	198	331300
3	3	1	3	1	2	A	A	1	198	332400
2	3	1	3	1	2		A	1	199	332400
2	3	1	3	1	2		A	1	198	332400
2	3	1	3	1	1		A	2	198	332400
3	3	1	3	1	1	A	A	2	200	332400
1	3	1	3	1	1		A	1	165	332400
3	3	1	3	1	1		A	2	165	332400
3	3	1	3	1	0	A	A	2	165	332400
1	3	1	3	1	0		A	2	165	332400
2	3	1	3	1	0		A	2	165	332400
3	3	1	3	1	0		A	2	168	332400
3	3	1	3	1	0		A	2	172	332400
2	3	1	3	1	0		A	1	163	332400
2	3	1	3	1	0		A	1	167	342400
4	3	1	3	1	0	A	A	1	170	342400
2	3	1	3	1	0		A	1	165	332400
2	3	1	3	1	0		A	1	166	332400
3	3	1	3	2	0	A	A	1	167	332400
2	3	1	3	2	1	A	A	2	166	332400
4	3	1	3	2	1	A	A	1	178	332400
4	3	1	3	2	1	A	A	1	192	332400
4	3	1	3	1	1	A	A A	1	186	332400
3	3	1	3	0	1		A A	1	189	332400
3	3	1	3	0	1		A	1	192	332400

Figure 2.6: Performance Analysis of Sampling Monitor (EYE) Measurements (S13)

A refinement of this type of monitor was later developed with the PP sampling monitor KIK (L5) which recorded data on the frequency and duration of system (PP) programs, and individual channel activity. The frequency of PP program calls was used to determine which system programs are most frequently called and should therefore be made resident in main memory, while less frequently called programs are stored on the system disc. A similar analysis was carried out on IBM/360 systems using the sampling monitor, CUE (B12). The frequency of occurrence of supervisor routines (SVCs) was measured, and the most commonly used routines were made resident in core.

Rodriguez-Rosell and Dupuy have described how a virtual machine dedicated to data collection was implemented on the CP-67 system (R1). The monitor communicates with the virtual machine supervisor, CP, at regular intervals and CP passes to it a snapshot of the system together with some cumulative and some time-averaged variables. This enabled the installation to determine how page fault rates, page stealing rates and CPU utilisation varied with the mean multiprogramming level, and hence to determine situations under which thrashing occurred.

2.4.5 Performance Analysis using Event driven Monitors

Event driven monitors are capable of gathering data of a more detailed nature than sampling monitors, although the cost of doing so may well be higher. The level of detail can be such that it is possible to make a detailed and accurate reconstruction of the sequence of events which occurred during a monitored period. Internal monitors can thus be used for giving a very detailed view of the performance of a system.

Examples of event driven monitors which can gather data at a very fine level of detail are the University of Texas CDC 6000 system internal monitor (S4), the Generalised Trace Facility (GTF) provided by IBM for the OS/360 and OS/370 systems (I2), and VM Monitor for the IBM 370 Virtual Machine operating system (C2). An event trace (i.e. the formatted output of the monitor, see figure 2.7),can be very useful for determining inefficiencies in system algorithms, or the cause of unusual situations, for example, system hangups or poor response times at certain times of the day. A good example of how an event driven monitor was used for this purpose is given by Callaway (C3).

Designers of operating systems, as well as their users, are interested in measuring the time taken to execute frequently called supervisor modules, with the objective of improving operating system performance. One of the performance tools incorporated into the Multics system (S1) was a package which recorded the time spent executing selected frequently called supervisor modules. For each selected module, the package recorded the number of times the model was invoked and the total execution time accumulated within each of a number of ranges of execution times. Use of this package resulted in the identification of inefficient supervisor modules and led to a redesign of some of the modules.

2.5 Performance Analysis of the Controlled Environment

2.5.1 Improving Computer System Performance

When a change is made to some part of the operating system in an attempt to improve performance (e.g. changing a scheduling algorithm),or when a system configuration is enhanced,it is desirable to determine what change in performance has actually taken place.

	CPU Changes	RA+1 Calls Del.Stk.Ent.	PP1	PP2	PP3	PP4
0	08.41.37.	3	4			
1			0	0	06 000 9958	0
2	3CPI 9958		LOD 4		CIO 3	
3	3CPO 9958		LOD 4		CIO 3	
4		1 1DB 0	LOD 4		CIO 3	
5			0	0	02 001 9960	0
6			LOD 4	1DB 1	CIO 3	
7			0	02 013 9966	0	0
8		3 CIO 9977	LOD 4		CIO 3	
9			LOD 4		CIO 3	CIO 3
10			0	0	0	02 014 9981
11			0	0	0	03 014 9983
12			0	0	0	06 003 9986
13	3CPI 9986		LOD 4	1DB 1	CIO 3	CIO 3
14	3CPO 9987		LOD 4	1DB 1	CIO 3	CIO 3
15			15 000 10003	0	0	02 003 9989
16			12 000 10005	0	0	0
17		3 RCL 10017		1DB 1	CIO 3	CIO 3
18		3 10		1DB 1	CIO 3	CIO 3
19	0IDL 10017			1DB 1	CIO 3	CIO 3
20	4NEW 10017			1DB 1	CIO 3	CIO 3
21		4 LDRP10018		1DB 1	CIO 3	CIO 3
22	0IDL 10018			1DB 1	CIO 3	CIO 3
23			0	03 013 10022	0	0
24			LDRP4	1DB 1	CIO 3	CIO 3
25			16 000 10024	12 000 10024	0	0
26	3NEW 10029		LDRP4		CIO 3	CIO 3
27		3 RCL 10030	LDRP4		CIO 3	CIO 3

Figure 2.7: Event Trace Generated by Event Driven Monitor (S4)

Measuring performance before and after the change is an obvious means of doing this. Sometimes the improvement in performance is so great that a detailed analysis is not required. More often, the change in performance will vary with the workload and performance may be improved in some circumstances and degraded in others. Such situations are difficult to analyse in the normal production environment, although statistical methods for doing this will be discussed in section 2.6. Alternatively, it is possible to virtually eliminate the fluctuations that occur in the real environment, by using a controlled environment, in which a model of the workload is applied to the computer system. The same tools may be used for measuring the performance of the controlled environment as are used for the production environment.

2.5.2 Analysis of the Batch Controlled Environment

The most common methods of creating a batch controlled environment are by means of benchmark or synthetic programs. As pointed out in Section 2.3.2, it is very important that the model of the workload be calibrated. Joslin and Aiken have clearly pointed out the dangers of using unrepresentative benchmarks with a revealing example (J2).

Using a model of the workload provides a very convenient means of comparing different versions of the operating system. In addition it may be used for evaluating entirely different computer systems. This is a frequently used method in computer system selection (G3, T3).

Using a batch benchmark which modelled the short job subset of the CERN workload. the author experimented with a Peripheral Processor and Channel Scheduling mechanism which had been incorporated into the CDC 6000 system at CERN (G5). He showed that the new scheduled system

functioned effectively in situations of low and high I/O activity and provided a more balanced and effective scheduling of I/O resources than the old unscheduled system. For the performance analysis, the CDC Dayfile was used to provide workload data, and the PP sampling monitor KIK was used to collect performance data.

2.5.3 Analysis of the Interactive Controlled Environment

With an interactive system a controlled environment may be created by using scripts (which are basically interactive benchmarks) which represent a user terminal session from login to logout (G6). Experiments may be carried out by keeping the workload constant and varying system parameters or algorithms. Alternatively, the system parameters may be kept constant and the workload varied in a measurable fashion. For example, the effect of increasing the workload on throughput rates, response times, CPU utilisation, etc. and the onset of system saturation may be determined. If a batch workload is also run, then the effect of increasing the interactive workload on the performance of the batch workload may also be determined (D4).

Gomaa and Lehman (G6, L2) have described how a performance tool called the Stimulator was used to apply a controlled interactive workload to a CDC 6000 Kronos system. Each Stimulator test consisted of a fixed number of simulated terminals executing pregenerated scripts. The interactive workload was increased in a measurable fashion by increasing the number of simulated terminals, and the performance of the system processing the controlled workload was measured using the CDC Dayfile. By this means, the onset of system saturation, with degradation in throughput rates and response times, was detected and the cause of the degradation analysed.

A tool such as the Stimulator has the additional advantage in that it is capable of simulating a larger number of terminals than actually exist on the system. It may thus be used for predicting the effect on performance of a greater interactive workload than is actually supported by the system in the production environment.

2.6 Performance Evaluation using Modelling Techniques

2.6.1 Application of Regression Analysis to Computer System Performance Evaluation

Bard used regression methods to analyse performance data collected by monitoring an IBM 360/67 running under the CP-67 operating system (B1, B2). Regression techniques were used to analyse 'CP overhead', that is the time spent by CP-67 in servicing user requests for system resources of various kinds, e.g. CPU time, main memory and various types of I/O operations. A multiple linear regression model was used to relate the average CP overhead time (dependent variable) to each of the system functions (independent variables) carried out by CP, so that the fitted regression coefficients provided estimates of the average CPU time spent servicing user functions by different parts of CP.

Watson (W4) applied regression techniques to the analysis of accounting data, and in particular to evaluate the change in performance due to the addition of 256K core memory to a 360/65 computer system. Regression models were constructed relating different performance measures (dependent variables) such as average CPU utilisation and average number of jobs processed per hour to various workload characteristics (independent variables). One of the independent variables was a dummy variable which was used to estimate the effect of the additional memory.

Waldbaum (W1, F3) used regression analysis techniques for evaluating changes made to an APL system running on an IBM 360/91 under OS/MVT. Multiple linear regression models were built for a number of points on the cumulative distribution function of the system response time. The models were used to evaluate the effect on system response time of reducing the average level of multiprogramming and increasing the maximum workspace size on disc.

2.6.2 Application of Trace Driven Simulation Modelling to Computer System Performance Evaluation

Simulation is the modelling technique which has been used most frequently in computer system performance evaluation. A simulation model is usually more realistic than an analytical model, and is not constrained by the usually artificial assumptions made for analytical convenience (G7). The most interesting results in simulation modelling have been obtained using trace driven techniques (C4) to represent the workload.

A trace driven simulation model of a CDC 6400 system running under the SCOPE 3.2 operating system was developed by Noe and Nutt (N3). The output of the model for each job consisted of predictions of the job's progress through the system. It is claimed that the model was able to satisfactorily predict the performance of the system under two extremes of workload situation, representing high and low percentages of short jobs. The model was written in Fortran and is said to be 50 times faster than the real system.

Waldbaum and Beilner developed a trace-driven simulation model of an IBM 360/91 running under the OS/MVT-LASP system (W2). The model was written in PL/1 and is said to process a whole day's computer activities in less than one minute of CPU time. Two interesting features of the model were:

a) Submodel Simulation

Submodelling is a means of structuring the model by representing only part of a system or running only part of a model, namely the submodel. This is achieved by replacing the internal information transmitted to a submodel, from other parts of the model, with input information (W3).

b) The Calibration Methodology

Calibration of the model is described as the process of tuning certain calibration parameters and changing parts of the model's structure so as to yield a good match between the model output and the real world output for a selected set of input data. Multiple linear regression techniques were used during the calibration process (B5, B7).

2.7 Conclusions

This chapter has surveyed the different tools and environments for the measurement and evaluation of computer system performance.

The evaluation of the three environments described in this chapter is not mutually exclusive, but rather complementary. If a model of the computer system is capable of modelling the system's performance in a fraction of the real world time, it is then possible to experiment with a much wider range of system and workload situations. Since a model is only an approximation to the real system, the most promising predictions of the model (e.g. parameter settings or scheduling algorithms) should be tried out on the real system. Experimental versions of the system may be prepared which can be run in a controlled environment where a calibrated model of the workload (using benchmark

or synthetic programs) is applied to the different versions of the system. Having identified the most promising version of the system in the controlled environment, the system changes may now be incorporated into the production system. The performance of the production environment before and after the changes should be monitored to enable the difference in performance to be evaluated. Regression analysis techniques may be used to estimate the change in performance in the normal production environment by separating out the effect, on performance, of the workload from the system modification.

CHAPTER 3: MODELLING OF COMPUTER SYSTEM PERFORMANCE

3.1 Introduction

A performance model of a computer system is an abstraction of the real computer system behaviour. This chapter describes the main aspects of computer system modelling and how they may be applied to the evaluation of computer system performance.

Section 3.2 describes the main features of computer system modelling. Section 3.3 describes three of the main techniques of modelling computer systems: queuing, regression and simulation techniques. Methods of structured modelling of computer systems are described in section 3.4. Section 3.5 describes methods of combining simulation and regression techniques in a multilevel hybrid modelling approach to the modelling of computer systems. Finally, section 3.6 introduces a particular application of the concepts presented, namely the modelling of the Imperial College CDC 6000 Kronos system.

3.2 Computer System Modelling

A computer system model to be used for the evaluation of computer system performance abstracts the behaviour of the real computer system, that is the behaviour of the computer hardware, together with the operating system. As input, the model requires an abstraction of the workload. As output, the model produces a record of predicted computer system behaviour.

Model development is a complex process. There are a number of steps involved in developing a model of a computer system before it can be used for experimentation.

(a) Understanding of the system to be modelled.

(b) An abstraction process. This involves deciding on the level of detail to be included in the model.

(c) Logical design of the model.

(d) Implementation. This involves producing a working model.

(e) Calibration. This process aims at reducing the behavioural differences between the real and modelled worlds, by making changes to the model (B4).

(f) Validation. This process aims at determining the domain of situations for which the model performs with a given accuracy, for an established calibration (B4, B6).

(g) Experimentation. This involves using the model in experimental situations which are different from those used during calibration and validation. For example, the model may be used to experiment on the effect of changes to the workload, system algorithms or parameters.

These stages of model development are likely to be iterative. For example some deficiency in the model may be discovered during validation. This may be due to a logical design error in the model, which may in turn be due to the system not being fully understood. Consequently, this may require a correction to the design error, modifications to the implementation, and a further calibration, prior to resuming the validation of the model.

3.3 Modelling Techniques

3.3.1 Introduction

A performance model of a computer system is an abstraction of the real computer system behaviour. This abstraction may be in the form of a mathematical model, which is a mathematical representation of the system. Queuing models and regression models are two examples of mathematical models. Alternatively, the abstraction may be in the form of a simulation model, which is an algorithmic representation of the system relecting system structure and logical procedure.

A model of a computer system may be static, in which case it may omit the recognition of time altogether, describe a snapshot of the state of the system at a moment in time, or model a steady state situation. A regression model is an example of a static model. On the other hand, a model may be dynamic, that is it may explicitly recognise the passage of time. A simulation model is an example of a dynamic model.

3.3.2 Queuing Models

Queuing models are analytical models of computer systems. In an analytical model, it is possible to deduce a solution to the problem under study directly from its mathematical representation (F1).

Queuing models of total computer systems usually involve a number of simplifying assumptions which make the model more amenable to mathematical analysis. The most common assumption is that the probability of getting a new request does not depend on how long ago the last request was made, sometimes called the "memoryless" property (G7). As a consequence of this assumption, the request inter-arrival time distribution follows an exponential distribution, which assigns the highest probability density to the smallest time interval of length zero. In many computing

environments, short values of inter-arrival times between requests are unlikely, since these requests are often due to human activity, and therefore the assumption is not valid (G7). A further simplification sometimes used is that of only analysing the steady state environment, and of making the assumption that the workload does not vary with time of day.

These assumptions tend to reduce the validity of queuing models in computer system performance evaluation. Queuing models have probably been of most use in modelling subsystems, e.g. CPU and memory management (C5), and I/O scheduling. Many analytical models have been developed with the objective of gaining insight into the system being modelled, rather than to evaluate system performance. Nevertheless, a few examples do exist of queuing models of total computer systems which have attempted to relate the model predictions to actual system performance (H2, W7).

3.3.3 Regression Modelling

3.3.3.1 Regression Analysis

Regression analysis is an empirical (i.e quantitative) method for analysing workload and performance data. Like many other empirical methods, it is a statistical method of analysing data (G7).

In a large computer system, there are likely to be many variables which are related to each other in some manner and whose quantities are continually changing. Often the functional relationship that exists between these variables is unknown or is too complicated to be described in simple terms. Regression analysis provides a means of approximating to this complex relationship by some simple mathematical function, such as a polynomial, which contains the appropriate variables and approximates to the true function over some limited ranges of the variables involved (D5).

A regression model deals with the following problem. Given a set of data containing the observations for several input (independent) variables X_1, X_2 ... X_k and an output (dependent) variable, it is required to fit the data by means of the function:

$$Y = f(X_1, X_2 \ldots X_k)$$

The function may have no physical meaning, but it may still be a valuable means of estimating the value of the dependent variable given the values of the independent variables (D5).

The functional relationship could be realised by means of a linear model:

$$Y = a_0 + \sum_{i=1}^{k} a_i X_i \qquad (1)$$

where a_i, $i = 0, 1, \ldots k$ are unknown parameters.

Given a set of m observations, where each observation consists of one set of values of all the variables in the model:

$$(Y_j, X_{1j}, X_{2j} \ldots X_{kj}) \quad j = 1,2, \ldots m$$

then the parameters a_0, a_1 ... a_k may be estimated by means of least squares fitting techniques. These parameters are called the regression coefficients.

A regression model of a computer system consists of an equation similar to equation (1). Once the regression coefficients have been determined, the model may be used for predictive purposes. A vector representing a set of values of the independent variables (X_1 ... X_k) is input to the model. The model then computes the predicted value of the dependent variable from equation (1).

3.3.3.2 Regression Modelling of Computer Systems

A regression model for computer systems performance may be shown diagramatically as in figure 3.1. A workload x, which consists of demands for the use of system resources (e.g. CPU, I/O devices, memory, etc.) is applied to the system (S2). The state of the system θ has two components

$$\theta = (\theta_1, \theta_2)$$

where θ_1 describes the hardware configuration and θ_2 is a set of control (tuning) parameters. The model predicts the performance of the system y, according to the functional relationship:

$$y = f(x, \theta)$$

If, for a particular set of experiments, the state of the system is constant, that is there is no change to the hardware configuration and no change to the control parameters, then the functional relationship reduces to:

$$y = f(x)$$

3.3.4 Simulation Modelling

3.3.4.1 Discrete Event Simulation

A computer system simulation model models the system's real world behaviour by means of an algorithmic abstraction of the system, reflecting system structure and logical procedure. A simulation model is dynamic, that is it explicitly deals with the passage of time, and simulating a system provides a means of studying the behaviour of a system over a period of time.

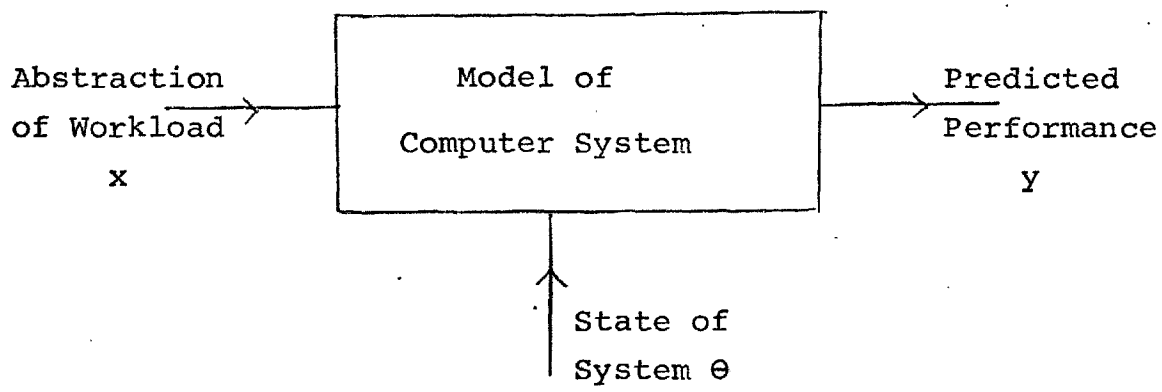


Figure 3.1: Regression Model of Computer System Performance

In a discrete event simulation model, changes of states in the system are represented by a collection of discrete events (F1). Changes of state only take place when an event occurs. Since the states of entities remain constant between events, the simulation is able to skip over the time between events. Consequently, at a moment in simulated time when an event occurs, the appropriate state changes are made to the model. Then simulated time is advanced to the next event, and the process is repeated. It is by this means that a simulation model is able to 'compact' time, and is thus capable of modelling a computer system's performance in a fraction of the real world time.

3.3.4.2 Computer System Simulation

A computer system simulation model models the behaviour of the real computer system, that is the computer hardware together with the operating system. As input, the model requires an abstraction of the workload. As output, the model produces a record of estimated computer system behaviour (figure 3.2).

Workloads have sometimes been modelled using probability distributions. However these often make unjustified assumptions about the workload. Alternatively the workload may be modelled by means of an event trace which is a set of workload characteristics obtained by monitoring the actual computer system processing the normal production workload (C4). Each job processed by the system is represented by a vector, which identifies certain characteristics of the job. The vector is input to the model at the simulated time of job arrival.

3.4 Structured Modelling of Computer Systems

3.4.1 Problems in Modelling Computer Systems

A number of factors have to be considered in constructing a model. In particular, factors of prime

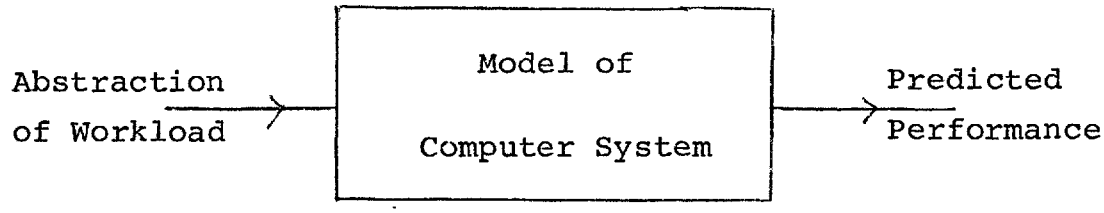


Figure 3.2: Simulation Model of Computer System Performance

importance are the cost of developing a model, the level of detail incorporated in the model, the speed of the completed model and its degree of accuracy.

The greatest drawback to simulation modelling is probably its relatively high cost (L6). This is closely linked to the problem of selecting the right level of detail to be included in the model. If the level of detail is too gross, the model may be unrealistic because important details may have been aggregated to such an extent that their effect is lost. On the other hand, if the level of detail is too fine, the model may be too expensive to use. A simulation model of a computer system has been described which was so complex that one minute of simulated time required 20 minutes of simulation (N1). For some experiments, but not all, this level of detail may be required.

In general, the more detail included in a model, the more closely is the model likely to represent the real world environment, and the more slowly is the model likely to function. Thus, a gain in the speed of the model is likely to be at the expense of its realism.

3.4.2 Multilevel Modelling

A more satisfactory approach is to model the computer system at several levels of detail. At each level, a self-contained model of the system is designed, implemented, calibrated and validated. At the next level, the model is refined further by adding more detail. Such an approach is called multilevel modelling.

Zurcher and Randell pioneered multilevel modelling (Z1) as a method of modelling a computer system design as it evolved by the systematic expansion of detail; by this means evaluation could be made an integral part of the design process. With this approach, the model may represent

what is happening in any part of the system without necessarily representing how it is happening. At each level of the model, greater detail may be introduced by gradually replacing what is happening in the model by how it is happening.

3.4.3 Advantages of Multilevel Modelling

In computer system evaluation, there are a number of advantages in adopting a multilevel modelling approach over the approach of developing a model at a single level of detail.

(a) Only as much detail need be incorporated into the model as is required for the level of information required and the aspect of the system under study. The level of information required by the installation manager,who may be interested in overall trends in the workload,is very different from the systems programmer who might be interested in optimising the memory scheduling algorithm. The requirement for refining the model to a further level by including more detail may be due to the desire for greater accuracy through a more realistic representation of the system,or the desire to experiment with lower levels of the system.

(b) In general, the less detail included in the model, the more the saving in time and cost to build, calibrate and validate the model and the more economical the running of the model becomes. On the other hand, the level of realism will be less.

(c) Experimenting with each level of the model is likely to provide significant insight into and understanding of the system. This will assist the model builder in designing, calibrating and validating lower levels of the system.

(d) The amount of workload data collected for input to the model and performance data for calibration purposes need only be what is required for the level of the model currently being implemented. Thus the quantity (and accuracy) of the data being input to and collected by the model should increase as the level of detailed representation increases.

3.4.4 Submodelling

Submodelling (W3) is a means of structuring the model into component parts, termed submodels. Each submodel has the property that it can run as part of the main model, receiving inputs from and feeding its output to other parts of the model. Alternatively, the submodel may run independently of the remainder of the model, in which case the inputs it would normally receive from other parts of the model are replaced by predefined trace or statistical data.

The advantages of submodelling are:

(a) The model may be calibrated more readily and possibly more accurately. Each submodel may be calibrated separately and independently of other submodels.

(b) The validation of the model can be carried out more readily and possibly more accurately. Each submodel of the model may be validated separately.

(c) Errors produced by various parts of the model may be estimated more accurately.

(d) Experiments involving only part of a system may be performed more efficiently and more accurately using the appropriate submodel.

3.5 Structured Modelling of Batch Computer Systems

3.5.1 Introduction

In this section, the application of the modelling methods described in the previous two sections to the modelling of batch computer systems is considered. This involves using regression and simulation modelling techniques in a hybrid simulation/regression model. It also involves applying multilevel modelling and submodelling methods to assist in modelling a computer system at several levels of detail.

3.5.2 Performance Measures in Batch Computer Systems

Before considering in more detail how modelling methods may be applied to the performance evaluation of batch computer systems, it is valuable to consider the alternative measures of batch system performance.

In batch computer systems, throughput, turnaround time and availability have been identified as three prime measures of performance (C1). Turnaround time is usually defined as the time between a user submitting his job at a computer reception to the time he receives his output. The main steps involved in this process are listed below and shown diagramatically in Figure 3.3.

(a) The user submits his job to a computer reception.

(b) The job is read through the card reader and enters the Input Queue.

(c) The job is scheduled for execution. During execution, the job's output is spooled to disc.

(d) The job terminates.

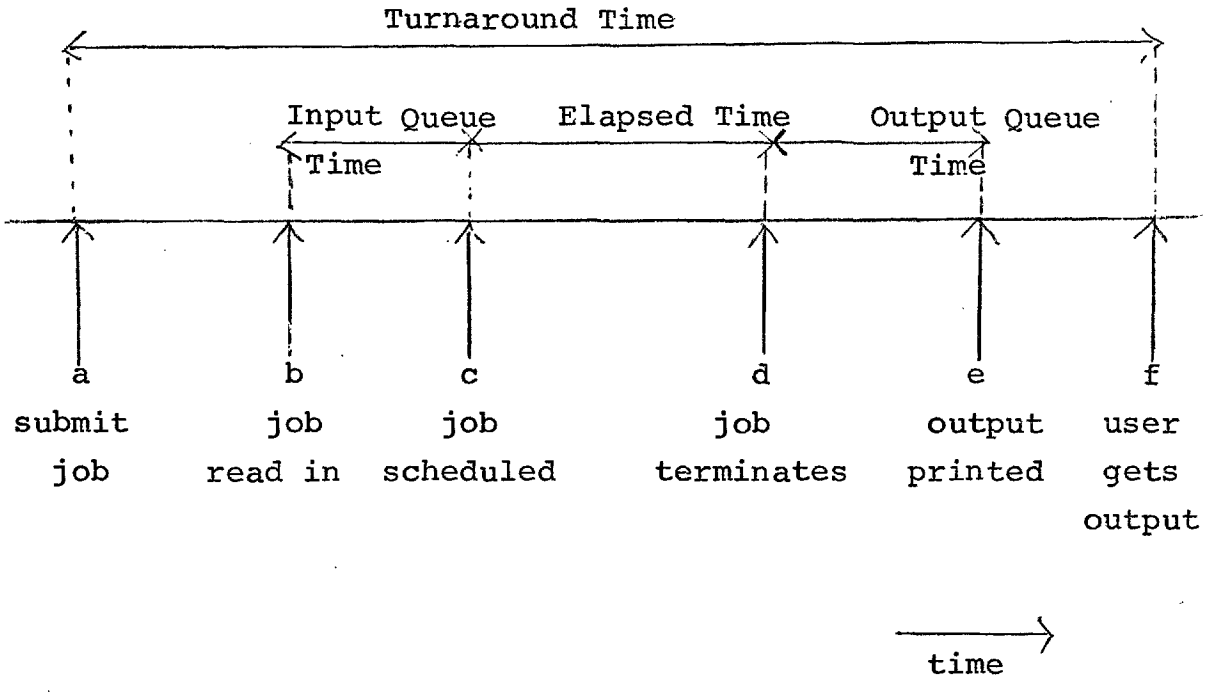


Figure 3.3: Steps in Batch Job Processing

(e) The job's spooled output is printed by a line printer.

(f) The output is returned to the user.

As defined previously, the turnaround time is the time between event a and event f. This definition of turnaround time suffers from the fact that the times between events a and b and events e and f respectively, are dependent on operator intervention. As a performance measure, it is more valuable to consider the time between event b and event e, that is the time when the job is directly under control of the computer system.

A batch job processed by a computer system passes through three sequential phases:

(i) Input phase. This is the period spent by the job in the Input Queue, that is the time between event b and event c.

(ii) Execution phase. This is the period when the job is in execution and competing with other jobs for physical resources. It is the time from when a job is scheduled for execution (event c) to the time it terminates execution (event d). This time is referred to here as the elapsed time of the job.

(iii) Output phase. This is the period spent by the job in the Output Queue, that is the time between event d and event e.

The prime measure of batch system performance to be used from now on in this thesis is the job elapsed time. The reasons for this are:

(a) It is a measure of performance which is appreciated by computer system users, installation managers, systems analysts and systems programmers.

(b) Elapsed time is readily measurable on most computer systems. In particular it was readily measurable on the system under study.

(c) It is a convenient measure in the modelling of computer systems at the job level, as it can be related directly to the characteristics of individual jobs, as well as to the load on the system when the job was run.

3.5.3 Multilevel Modelling of Batch Computer Systems

In applying a multilevel modelling approach to a batch computer system, let us first consider the highest level model. It seems logical at this level to model workload performance. The performance of batch jobs in the system may be described entirely by means of a regression model which models what happens to jobs when processed by the system.

In the previous section, job elapsed time was chosen as the prime measure of batch system performance. A regression model of a batch computer system may be developed in which the dependent (i.e. output) variable is the job elapsed time. The independent (i.e. input) variables are measures of the job's resource demands. Once such a model has been calibrated and validated, it would then be capable of predicting a job's elapsed time, given the job's resource demands.

More detail may be introduced at the next level, either by carrying out a more detailed regression analysis, or by introducing simulation techniques. One method of

performing a more detailed regression analysis is by modelling job step performance. A regression model could be built which predicts job step elapsed time, given the job step resource demands. Furthermore, different regression models could be built for different types of job steps. For example, self-contained regression models of the compilation, link/loading, execution and utility job steps could be developed. For a given job, the sum of the predicted job step elapsed times is then the predicted job elapsed time.

More detailed modelling may be achieved by further regression modelling. Alternatively, simulation techniques may be introduced and combined with the regression techniques to form a hybrid model.

3.5.4 Hybrid Computer System Modelling

3.5.4.1 The Framework for a Hybrid Simulation/Regression Model

A regression model is static, i.e. it does not recognise the passage of time. A simulation model is dynamic and is capable of modelling system structure and logical relationships. It also allows the interaction between jobs competing for limited resource to be dynamically modelled. A hybrid model should be dynamic, although capable of using the static features of a regression model.

One method of constructing a hybrid simulation/regression model involves using a regression model whose dependent variable is in units of time. In the previous section, a regression model was suggested which could predict a job's elapsed time, given the job's resource demands as input. To produce a dynamic model, this regression model is incorporated within the framework of a hybrid simulation/regression model. Within this frame-

work, the regression model of job elapsed time in effect becomes a regression submodel, since it is now a self-contained part of the overall model.

A co-ordinating routine is introduced to provide the simulation framework. In its simplest form, the hybrid simulation/regression model is shown in figure 3.4. The co-ordinating routine inputs a vector representing the job's resource demands to the regression submodel at the simulated time of job arrival. The regression submodel predicts the job's elapsed time. The co-ordinating routine adds the predicted elapsed time to the simulated time of job arrival to give the predicted time of job termination. Thus by imbedding the regression model within a simulation framework, the static regression model is converted into a dynamic simulation/regression model.

3.5.4.2 Increasing the Level of Detail of the Hybrid Model

Once the framework for incorporating a regression submodel within a hybrid simulation/regression model has been set up, the level of detail of the model may be increased using the same framework. Increasing the level of detail may be accomplished either by means of further regression modelling, further simulation modelling or both.

Using further regression modelling, regression submodels of job step performance may be developed, as mentioned in the previous section. One or more of these job step submodels may be linked together by means of a co-ordinating routine. A regression submodel is invoked at the time of job step arrival to predict the job step elapsed time. This estimate is then added on to the simulated job step arrival time to give an estimate of job step termination time.

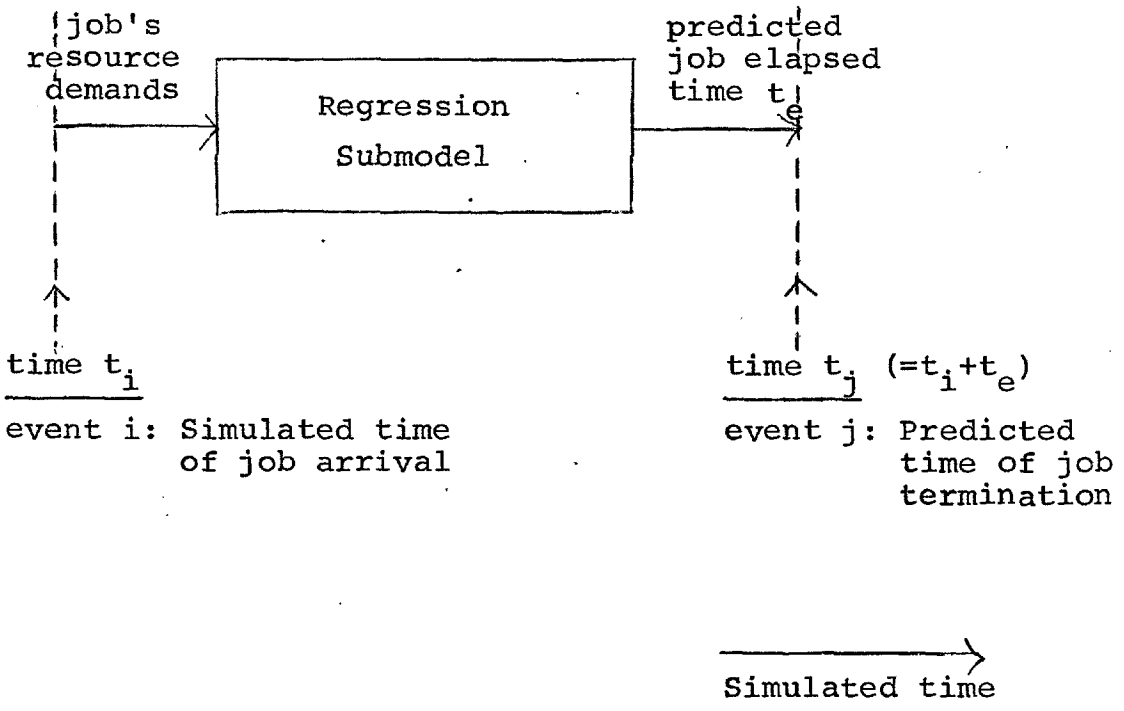


Figure 3.4: A Simple Hybrid Simulation/Regression Model

As an example consider a hybrid simulation/ regression model with three regression submodels, a compilation job step submodel, a link/load step submodel and a job execution step submodel. Consider the passage of a job with three job steps, compile, load and execute, through the system (figure 3.5). At the simulated time of job arrival,the co-ordinating routine inputs a vector representing the job's compilation resource demands to the compilation submodel, which predicts the job's compilation time. The co-ordinating routine adds this estimate to the simulated time of job arrival to give the predicted time of job step termination. Next, the co-ordinating routine inputs a vector representing the job's link/load resource demands to the linker/loader submodel, which predicts the linking/loading time. Finally, the job execution step submodel is invoked to predict the job execution time. The predicted time at the end of this job step is then the predicted time of job termination.

3.5.4.3 Simulating Subsystems within the Hybrid Model

Using further simulation modelling, different subsystems of the system may be simulated in more detail. By this means system structure and logical procedure may be introduced into the model. Furthermore, algorithms used in the actual system may be simulated. For instance the simulation of job and memory scheduling could be introduced into the model. Such a model may only schedule a job for execution if sufficient memory is available for it.

Consider the example of a multiprogramming system with fixed size non-relocatable partitions as in IBM OS/360 MFT. The algorithms for job and memory scheduling in such a system may be incorporated into the model such that a job is only scheduled into an appropriate partition when a partition becomes free. Once a job has been scheduled into main memory, a regression submodel (or a series of regression

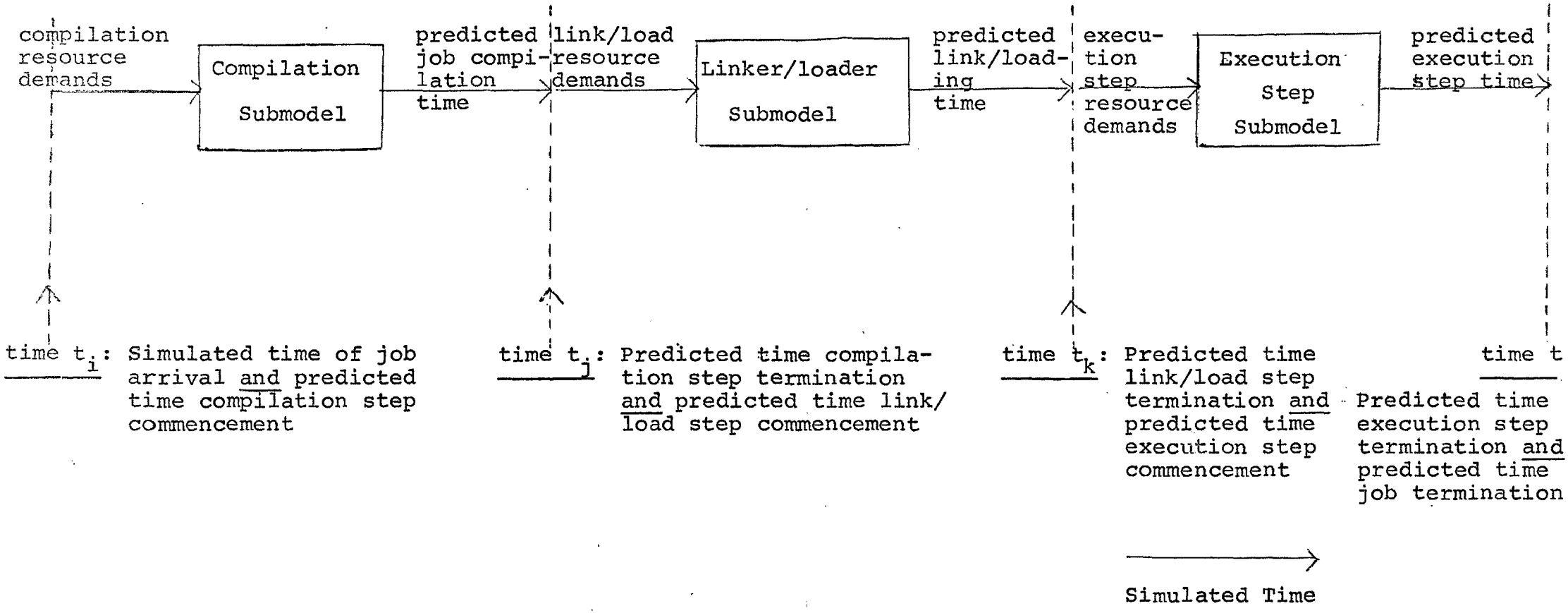


Figure 3.5: Hybrid Model with Three Regression Submodels

submodels) is invoked, as before, to predict the job elapsed time. Clearly, this method may be extended to include different memory scheduling algorithms and the simulation of different subsystems.

3.6 Multilevel Hybrid Modelling of the Imperial College CDC 6000 System

The concepts presented in the previous sections have been applied to the modelling of an actual computer system, namely the Imperial College (IC) CDC 6000 Kronos system. The system has been modelled at three levels of detail:

Level 1: The Workload Model
Level 2: The Load Adjusting Model
Level 3: The Memory Management Model

Each level consists of a self-contained model of the system. The models are fast approximate models of the execution phase of a batch job, that is from the time a job is first scheduled for execution to the time it terminates. The job elapsed time is used as the prime measure of performance.

The first level model uses regression modelling techniques entirely. A regression model is static and is therefore not capable of dynamically adjusting its estimates of the load on the system as each modelled job executes. By combining simulation and regression techniques, the second level hybrid model is capable of dynamically adjusting its estimates of system load. At the third level, more detail is introduced, within the hybrid framework, by simulating the memory management subsystem.

CHAPTER 4: THE IMPERIAL COLLEGE CDC 6000 KRONOS SYSTEM

4.1 Introduction

A Control Data (CDC) 6400 system was installed at Imperial College (IC) in 1971. The initial operating system used was SCOPE 3.3. In summer 1972, this was replaced by the Kronos 2.0 operating system. In the early summer of 1974, a newer version of the operating system, Kronos 2.1, was introduced. Later that summer, the Imperial College Computer Centre (ICCC) was considerably enhanced by the installation of a CDC CYBER 73 computer system. Apart from a few differences, the CYBER 73 is architecturally very similar to the 6400*.

This chapter describes the main features of the Imperial College system. Section 4.2 describes the structure of the IC workload. The main features of the CDC 6000 architecture and system configuration are outlined in section 4.3. An overview of the Kronos operating system is given in section 4.4. The job processing and memory scheduling aspects of the system are described in more detail in sections 4.5 and 4.6 respectively.

4.2 The Imperial College Workload

The CDC 6000 Kronos (C8, C9) system at Imperial College supports four types of service:

(a) A timesharing service is provided by means of interactive terminals linked to the Telex subsystem (C9) which operates under Kronos. Terminal users are able to create and edit files stored on direct access devices; enter, edit, compile and run programs; and submit batch jobs to the system for subsequent execution.

---

* When both machines are referred to together in this thesis, the term CDC 6000s will be used.

(b) A 'cafeteria' service is provided for short batch jobs. The service has a dedicated card reader and line printer situated in a room adjacent to the computer room.

(c) A local batch service is provided for all categories of batch jobs outside the smallest (cafeteria service) category. The turnaround time for batch jobs varies from a few hours to several days depending on the job category and the load on the system.

(d) A remote batch service is provided for batch jobs of all categories. This is provided by means of low speed remote job entry terminals linked to the Export/Import subsystem which operates under Kronos.

Batch jobs may fall into one of five job categories. The category a job is placed in depends on the resources required by the job. The resource limits for each category are shown in table 4.1.

Under Kronos 2.0, the CDC 6400 supported the batch, remote batch and timesharing workloads. With the introduction of the CYBER 73, the workload was divided between the two machines. Most of the batch work and all the remote batch work was run on the CYBER. The 6400 supported the timesharing workload as well as some batch work, mainly that generated by the terminal users. The two systems ran independently of each other.

In June 1975, the locally developed multi-mainframe software was introduced. This enables the two systems to support a shared permanent file base.

		Job Category				
		1	4	7	10	13
Resource	Central Memory	24K	25K	30K	40K	49K
	Lines Printed	768	1536	2048	3072	5120
	CPU time(secs)	16	65	180*	300*	600*
	Magnetic tapes	0	2	3	3	4
	Cards Read	768	2048	3072	3072	3072
	Cards Punched	1024	1024	3072	3072	3072

* In 6600 CPU seconds $\simeq$ 2.5x6400 CPU seconds

Table 4.1: Job Categories on the Imperial College CDC Kronos System

		Priority Level							
		7	6	5	4	3	2	1	0
Job Category	1	-	-	-	P	M	J	G	A
	4	P	O	N	M	J	G	D	A
	7	M	L	K	J	G	E	C	A
	10	J	I	H	G	D	C	B	A
	13	H	G	F	E	D	C	B	A

Key: Priority levels: P highest, A lowest

Table 4.2: Job Classes on the Imperial College System

4.3 The Imperial College CDC 6000 Computer System

The Imperial College CDC 6400 (C7) and CYBER 73 each consist of:

(i) A Central Processor which executes user jobs.

(ii) 64k of 60-bit word Central Memory (CM), in which user jobs are multiprogrammed.

(iii) 10 Peripheral Processors (PPs), each of which has its own 4k 12-bit private memory. The PPs perform all I/O tasks and most of the operating system functions. The PPs communicate with the CPU via CM.

(iv) 12 data channels. Any PP may read or write to any channel, under software control.

The main architectural difference between the 6400 and the CYBER 73 is the compare/move unit on the CYBER. This processes additional character handling instructions,which are unavailable on the 6400. These instructions are made use of by some of the compilers and run time I/O packages on the CYBER.

The 6400 also had 250k of 60-bit word Extended Core Storage (ECS) attached to it, which is used as a fast peripheral. When the CYBER was introduced, each machine was allocated 125k of ECS, which is not shared.

Both 6000 machines are equipped with several disc drives and magnetic tape units.

4.4 The Kronos Operating System

4.4.1 System Monitor

In early CDC 6000 operating systems, the Nucleus

of the operating system, called Monitor, resided entirely in a dedicated PP. In the Kronos system, the supervisory duties are divided roughly equally between the dedicated PP Monitor (PPMTR) and the Central Monitor (CPMTR). CPMTR resides in Central Memory and is executed by the CPU. In general, CPMTR provides a faster response to user requests than the entirely PP resident Monitor. In this paper, no further distinction will be made between CPMTR and PPMTR; reference will be made instead to Monitor (MTR).

MTR also controls the execution of system tasks by the 8 pool PPs. (The tenth PP is dedicated to driving the Dynamic System Display). Pool PPs are allocated tasks to execute by MTR. When a PP finishes executing a particular task, it returns to the pool. Some of the tasks are specifically requested by a user program, e.g. for I/O, in which case the PP program CIO is loaded to service the request. Other tasks are initiated by MTR to control job processing. The Job Scheduler, 1SJ is called to schedule jobs for execution and the Job Advancer 1AJ is called to initiate job and job step execution.

4.4.2 Control Points

In the Kronos system, Central Memory (CM) is divided into a system area and user area. The system area holds the Central Memory Resident (CMR), which consists of a number of system tables, and Central Monitor. User jobs are multiprogrammed in CM.

Each job in Central Memory is assigned to a logical entity called a control point. The control point is allocated various system resources to enable the job to be executed. This includes a contiguous block of CM in which the job resides. The Central Processor will be allocated to the control point from time to time to execute the job.

Peripheral processors will be allocated to a control point as required, to execute various tasks such as I/O and to perform system functions such as job and job step initiation. For each control point, there is a corresponding block in CMR, called the control point area, which holds information about the job.

The amount of CM allocated to a control point may change during execution, as memory is allocated at the job step and sometimes sub-job step level. Jobs may be relocated in CM to provide room for other jobs. A job may also be rolled out of Central Memory and forced to release its control point, if a higher priority job is scheduled in its place. ECS is used as a primary rollin/ rollout device with disc as a secondary device. When a job is rolled out, the entire contents of its control point are written out to the rollout medium, together with the content of its control point area. The job is placed in a Rollout Queue and will eventually be scheduled back into CM.

4.4.3 Subsystems

Some control points are used to carry out specific system functions. One control point supports a spooling program BATCHIO which uses its own dedicated PP to drive the slow peripheral devices. A second control point supports the interactive subsystem TELEX which uses a dedicated PP to drive the hardware multiplexor, to which the interactive terminals are attached. A third control point, supports the remote batch subsystem EXPORT/IMPORT which again uses a dedicated PP to drive a second hardware multiplexor, to which are linked the remote batch terminals.

4.5 Job Processing on the Kronos System

4.5.1 Batch Job Management

A batch job submitted to the system must be in one of five categories (table 4.1) depending on its resource requirements. In addition, a user must specify a priority level for his job, between zero, the lowest level and 7 the highest. The Imperial College developed Job Manager(ICQMAN) maps each job into a job class, based on its category and priority, as shown in table 4.2 (W5). All jobs, apart from J1 jobs input via the cafeteria service, are then saved in the permanent file base.

Every file in active use by the system (input, output, rollout and local files) must have an entry for it in the File Name Table (FNT), which is part of CMR. Before a job can be scheduled for execution by the Job Scheduler, lSJ, it must have an entry for it in the FNT. At regular intervals, ICQMAN retrieves jobs, depending on their job class, from the permanent file base and creates FNT entries of type Input for them. All files of type Input in the FNT constitute the Input Queue (which is not a linked list) and are then available to lSJ for job scheduling. Eventually a job will be scheduled to run at a control point.

4.5.2 Terminal Management

If an interactive user logged into the system is typing a program or doing some simple line editing, he makes use of facilities provided by the Telex subsystem, and thus only imposes a relatively small load on the system. On the other hand, if he wishes to compile or execute a program interactively, then a much larger demand is made on the system. For compilation or execution, an entry is created for the interactive user in the Rollout

Queue. This looks to the system like a job with one job step and is referred to here as a terminal job. The Rollout Queue consists of all entries in the FNT of type Rollout. All jobs in the Rollout Queue are available for selection by the Job Scheduler, 1SJ. Eventually a job in the Rollout Queue will be brought into CM to execute at a control point. Thus a terminal job must compete for system resources with other batch and terminal jobs.

4.6 Memory Scheduling

4.6.1 Job Priorities

In spite of its name, the Job Scheduler 1SJ deals with memory scheduling in addition to job scheduling. It therefore selects jobs for execution from both the Input (batch jobs) and Rollout (batch and interactive jobs) queues.

1SJ bases its decisions on the Central Memory priority of the job. There are a set of installation dependent priority parameters, namely the lower bound, upper bound and entry priorities, associated with each job origin. The job origin identifies whether the job originated from Batch, Telex or Export/Import.

A job may enter a given queue with an entry priority which is between the lower and upper bounds for that job origin. In that case, the job's priority is gradually aged until it reaches the upper bound. If a job enters a given queue with a priority outside the lower bound/upper bound range, then its priority is not aged. When a batch job is first entered into the Input Queue by the Job Manager (ICQMAN), it will be given a CM priority according to its job class. This priority may or may not lie within the aging range.

1SJ bases its decision on which job to schedule next, on a job's CM priority and the maximum memory required by the job, as specified on the job card. If necessary, 1SJ will roll out jobs of lower priority. Once a job is rolled in, it is allocated a CM priority equal to the upper bound priority of its queue and origin.

4.6.2 Job Time Slices

As the Kronos system is primarily a timesharing system, its algorithms are oriented towards providing a fair allocation of resources. Consequently, each job in CM is allocated two time slices, the values of which depend on the job origin. These time slices set an upper time limit on the use of two critical resources, CM and CPU, that a job may use while resident in CM. Once a job exceeds one of its time slices its CM priority is reduced, usually to the lower bound priority of the Roll-out queue for that job origin. The CM time slice is the real time that a job is allowed to execute in CM for before having its priority reduced. The CPU time slice is the amount of CPU time a job is allowed to use before having its priority reduced.

The CPU scheduling algorithm on the Kronos system is basically a simple round-robin scheduling algorithm.

CHAPTER 5: THE CDC DAYFILE ACCOUNTING SUBSYSTEM

5.1 Introduction

In CDC 6000 systems, accounting data is collected by the Operating System and stored in a disc file called the Dayfile. The contents of the Dayfile are dumped to magnetic tape at regular intervals by the operators. Analysis of the Dayfile can provide much information about the characteristics of the workload and overall throughput rates.

The Kronos Dayfile was used as the source of workload and performance data for the evaluation and modelling of the Kronos system described in this thesis. Section 5.2 describes the structure of the Dayfile. Section 5.3 describes the main aspects of the processing of the Dayfile. Section 5.4 describes the three periods monitored and the data collected. Section 5.5 describes the limitations of the Dayfile for performance evaluation and modelling.

5.2 Structure of the Kronos Dayfile

5.2.1 Kronos Dayfiles

The Dayfile is maintained by the system Monitor (MTR). Any system program (central or peripheral processor) may record a message in the Dayfile by making a request to MTR. MTR adds the current time to the message before recording it in the Dayfile. Dayfile messages are recorded in chronological order (G4).

In the Kronos system, more than one Dayfile is maintained. Two of the Kronos Dayfiles, the Account Dayfile and the System Dayfile, were used in this analysis. These two Dayfiles are maintained for the whole system, so that messages recorded in them relate to all batch jobs and time sharing users processed by the system. A

third Dayfile, the User Dayfile is maintained for each batch job. This contains all messages pertaining to that particular job. In Kronos, a system program must state which Dayfile(s) it wants a message to be recorded in.

5.2.2 User Dayfile

A User Dayfile is maintained for each job executing at a control point. Three types of messages are recorded in the User Dayfile.

(a) The job card and all control cards processed by the system for that job.

(b) Error and information messages relating to each job step executed.

(c) At job termination, data on the resource utilisation of the job is output.

The contents of the User Dayfile are printed with the job's output after job termination. An example of the User Dayfile is shown in figure 5.1.

5.2.3 Account Dayfile

This Dayfile records information which is mainly of use for job accounting. Information is recorded for both batch and time sharing users. An example of the Account Dayfile is given in figure 5.2.

For both the Account and System Dayfiles, in addition to recording the time the message was issued, MTR also records the job name of the job for whom the message was issued. The job name is a unique seven charac-

```
19.09.47.JOB(UMACH09)
19.09.50.TRAY NO.              HGI
19.09.51.COPYBR(INPUT,POSTPRC)
19.09.51.  COPY COMPLETE.
19.09.52.REWIND(POSTPRC)
19.09.52.REPLACE(POSTPRC=POSTMY)
19.09.55.FUN(S,I=POSTPRC)
19.10.14.CTIME  003.765 SEC.   MAY 1971.
19.10.14.LIBFILE(QUICAT)
19.10.18.QUICAT(F)
19.10.20.  QUICAT DONE.
19.10.21.RU          0.090 UNITS
19.10.21.CU          0.090 UNITS
19.10.23.CP          3.972 SEC.
19.10.23.CM          0.026 KWH.
19.10.23.MS          0.203 KPR.
19.11.04.LP20        0.749UMACH09    .008
```

Figure 5.1: Example of User Dayfile

ter name which is given to a batch job when it first enters the system, or is associated with a terminal session when a user first logs in (see figure 5.2). An eighth character is added onto the end of the job name to identify the origin of the job:

B - Batch job
T - Telex (terminal) user
E - Export/Import (remote job entry) job
S - System job

The data recorded for batch jobs in the Account Dayfile is as follows:

(a) time a batch job is read through the card reader, and the number of cards read.

(b) time a job commences execution. At this time, a copy of the user's job card is recorded in the Dayfile.

(c) time a job terminates.

The following job resource utilisation data is output at job termination:

(d) CPU time used.

(e) An estimate of Central Memory utilisation: the product of CM and CPU utilisation in units of Kiloword hours (KWH).

(f) Disc physical records input or output.

(g) Magnetic tape physical records input or output.

(h) Time the job's output is printed, together with the number of lines printed.

time message issued

job name

job origin

```
14.17.03.QDAAYAMB.  MS          0.050  KPR.
14.17.03.QDAAYAMB.  UN          0.004  UNITS
14.17.03.ATQQ031T.  USER  UMECP07   LOG OFF.
14.17.03.ATQQ031T.  CP         21.994  SEC.
14.17.03.ATQQ031T.  CM          0.093  KWH.
14.17.03.ATQQ031T.  MS          0.846  KPR.
14.17.03.ATQQ031T.  TTYO        1.312  CHARACTERS.
14.17.03.ATQQ031T.  TTYI        0.255  CHARACTERS.
14.17.04.QDACGAQB.  CR          0.313  KCD.
14.17.06.QDACGAQB.  ST  BC0034       SITE ADDRESS
14.17.06.QDACGAQB.  JOB(UMEEB13)                          <- Job Started
14.17.07.AESA054T.  USER  UMEMN05   LOGGED IN.            <- User logged in
14.17.08.QDACGAQS.  US    UMEEB13
14.17.08.QDACGAQS.   J          1
14.17.08.QDACGAQS.  SP          4
14.17.11.QDABKBMB.  ST  BC0077       SITE ADDRESS
14.17.11.QDABKBMB.  JOB(CMAAR59,J4,MT1) ULCC JOB
14.17.13.QDABKBMS.  US    CMAAR59
14.17.13.QDABKBMS.   J          4
14.17.13.QDABKBMS.  SP          4
14.17.13.QDABUBMB.  ST  BC0002       SITE ADDRESS
14.17.13.QDABXAQE.  LP          0.844  KLN.
14.17.13.QDABUBMB.  JOB(UMEMN05,J4,SP4,T24,CM25000,LC2000,MT
14.17.13.QDABUBMB.  ()       J.FITZPATRICK  NUC.POWER
14.17.14.QDABUBMS.  US    UMEMN05
14.17.14.QDABUBMS.   J          4
14.17.15.QDABUBMS.  SP          4
14.17.22.QDAAYAME.  LP          0.253  KLN.
14.17.25.QDACGAQB.  CP          4.336  SEC.                 } Job terminated
14.17.25.QDACGAQB.  CM          0.028  KWH.                 }
14.17.25.QDACGAQB.  MS          0.466  KPR.                 }
14.17.25.QDACGAQB.  UN          0.047  UNITS
14.17.25.QDACBAQB.  LP          0.150  KLN.
14.17.28.ALDI057T.  USER  UMEMH74   LOGGED IN.
14.17.34.AM3A012T.  USER  UMEMM62   LOGGED IN.
14.17.43.QDABKBMB.  VSN  ULLC1AF    ASSIGNED.
14.17.48.ALDI057T.  USER  UMEMH74   LOG OFF.
14.17.48.ALDI057T.  TTYO        0.198  CHARACTERS.
14.17.48.ALDI057T.  TTYI        0.026  CHARACTERS.
14.17.49.QDACAAQB.  LP          0.366  KLN.
14.17.56.QCECSAMB.  CP         23.103  SEC.
14.17.56.QCECSAMB.  CM          0.149  KWH.
14.17.56.QCECSAMB.  MS          0.536  KPR.
14.17.56.QCECSAMB.  UN          0.248  UNITS
14.18.02.AFFYUD2E.  LOGOFF  UMEAW01.
14.18.02.QDABUBMB.  CP          5.434  SEC.
14.18.02.QDABUBMB.  CM          0.037  KWH.
14.18.02.QDABUBMB.  MS          0.514  KPR.
14.18.02.QDABUBMB.  UN          0.061  UNITS
14.18.03.QDACHAQB.  CR          0.404  KCD.
14.18.03.QDACHAQB.  ST  BC0034       SITE ADDRESS
14.18.04.QDACHAQB.  JOB(UMCA052.
14.18.05.QDACHAQS.  US    UMCA052
14.18.05.QDACHAQS.   J          1
14.18.05.QDACHAQS.  SP          4
14.18.10.AWUO073T.  USER  ZHACC22   LOGGED IN.
14.18.11.QDABBAMB.  CP          1.345  SEC.
14.18.11.QDABBAMB.  CM          0.005  KWH.
14.18.11.QDABBAMB.  MT          0.167  KPR.
14.18.11.QDABBAMB.  MS          0.126  KPR.
14.18.11.QDABBAMB.  UN          0.039  UNITS
14.18.11.QDACDAQB.  LP          0.376  KLN.
14.18.13.AESA054T.  USER  UMEMN05   LOG OFF.             } User logged out
14.18.13.AESA054T.  CP          0.031  SEC.                 }
14.18.13.AESA054T.  MS          0.081  KPR.                 }
14.18.13.AESA054T.  TTYO        0.341  CHARACTERS.          }
14.18.13.AESA054T.  TTYI        0.034  CHARACTERS.          }
14.18.16.QDABKBMB.  CP          0.651  SEC.
14.18.16.QDABKBMB.  MS          0.036  KPR.
14.18.16.QDABKBMB.  MT          0.083  KPR.
14.18.28.QDACIAQB.  CR          0.254  KCD.
14.18.28.QDACIAQB.  ST  BC0034       SITE ADDRESS
14.18.29.QDACIAQB.  JOB(UMEMN18,J1)
14.18.31.QDACIAQS.  US    UMEMN18
14.18.31.QDACIAQS.  SP          4
14.18.31.QDACIAQS.  SP          4
14.18.35.AWUO073T.  USER  ZHACC22   RECOVERED.
14.18.38.QDACIAQB.  CP          0.077  SEC.
14.18.38.QDACIAQB.  MS          0.107  KPR.
14.18.39.QDACCAQB.  LP          0.583  KLN.
14.18.42.QDAADASS.  CP         35.841  SEC.
14.18.42.QDAADASS.  CM          0.175  KWH.
14.18.42.QDAADASS.  MT          0.274  KPR.
14.18.42.QDAADASS.  MS          1.904  KPR.
14.18.42.QDAADASS.  UN          0.348  UNITS
```

Figure 5.2: Example of Account Dayfile

For an interactive user the following data is recorded in the Dayfile:

(a) time user logged in

(b) time user logged out

The following resource utilisation data is recorded at user logout:

(c) CPU time used

(d) Product of CM and CPU utilisation in Kiloword hours

(e) Disc physical records input or output

(f) Characters input by user at teletype

(g) Characters output by system at teletype

5.2.4 System Dayfile

Messages recorded in the System Dayfile are of the following type:

(a) All batch job steps processed and the time of their initiation.

(b) All terminal commands executed at a control point and the time of their initiation.

(c) All job steps executed by system jobs and the time of their initiation.

(d) When Telex (the timesharing subsystem) is dropped at the end of a timesharing session, it outputs data to the System Dayfile on the characteristics of the session.

```
11.55.29.NYATIAQE.  PASSWOR
11.55.29.NYATIAQE.  GET(MYLIB)
11.55.31.NYATIAQE.  FUN(S)
11.55.31.NYATBAAL.  STOP
11.55.33.NYATIAQE.  CTIME 000.217 SEC.  MAY 1971.
11.55.33.NYATIAQE.  MAP(PART)
11.55.33.NYATIAQE.  LOAD(LGO,MYLIB)
11.55.34.NYATIAQE.  EXECUTE.
11.55.36.AAAA002M.  MSORT,L024.
11.55.38.NYATIAQE.  FATAL ERROR   84
11.55.38.NYATIAQE.  ABNORMAL TERMINATION
11.55.38.NYATIAQE.    84 ERROR DETECTED BY OUTPTC
11.55.44.A1Y1P46T.  &LOC,O,PLOTT,,2.
11.55.45.NYATJAQE.  JOB(UMEMM16,J1)
11.55.47.NYATJAQE.  PASSWOR
11.55.47.NYATJAQE.  GET(SAMARA)
11.55.48.AZWQ040T.  &PACK,AGFNJS.
11.55.48.AZWQ040T.   PACK COMPLETE.
11.55.49.A1Y1P46T.  PLOTT   --014400
11.55.50.NYATJAQE.  UPDATE(P=SAMARA,Q)
11.55.50.NYATJAQE.  READING INPUT
11.55.50.NYATJAQE.                1 UPDATE ERRORS, JOB ABORTED.
11.55.51.AOLY050T.  DISPOSE,RUNSAVE=PRINTSF.
11.55.57.AOYI0241.  &LPF,330,334,FAS.
11.55.58.AAAA002M.  MSORT,L072.
11.55.59.AMAA072T.  &LPF,&CD,REFORM.
11.55.59.AZWQ040T.  KREDIT,AGFNJS.
11.56.02.ALLY050T.  &CATLIST.
11.56.10.ALL4QB73T.  &LPF,HYHOHY.
11.56.13.AEOY034T.  COMPASS,1=INQIO.
11.56.13.AEOY034T.   FL TOO SHORT.
11.56.19.AAAA002M.  MSORT,L045.
11.56.20.ASOY0451.  MNF(1,D,K,)=BOSIDE)
11.56.20.AEOY034T.  REL,45000.
11.56.22.NYATMASS.  X=OSTATUS(O,J=NYAASBG)
11.56.22.NYATMASS.    OSTATUS(O,J=NYAASBG)
11.56.29.NYAASBGE.  JOB(UMEMM01,J10,T300,CM40000,LC5000)
11.56.31.NYATHAPE.  JOB( UMEA083,J1 )
11.56.31.NYAASBGE.  PASSWOR
11.56.31.NYAASBGE.  GET(OLDPL=PFNAME)
11.56.33.AF210051.  &CATLIST.
11.56.33.NYATHAPE.  PASSWOR
11.56.34.ASOY0451.    CTIME =     3669MS  PSR LEVEL 1
11.56.34.ASOY0451.    CORE NEEDED IS     0515668
11.56.35.AEOY034T.  COMPASS,1=INQIO.
11.56.35.AAAA002M.  MSORT,L024.
11.56.36.AEOY0341.   ASSEMBLY COMPLETE.
11.56.36.NYATHAPE.  MNF(E = B)
11.56.37.NYATLAQE.  JOB(ZMEMM58,J1)
11.56.37.NYATLAQE.   FELLORIN
11.56.39.NYATLAQE.  PASSWOR
11.56.41.NYATHAPE.    CTIME =     2330MS  PSR LEVEL 1
11.56.41.NYATHAPE.    CORE NEEDED IS     0532378
11.56.41.NYATHAPE.  MAP(P)
11.56.41.NYATHAPE.  B.
11.56.41.ADEY030T.  &PACK,LAD.
11.56.42.ADEY030T.   PACK COMPLETE.
11.56.42.NYATLAQE.  MAP(PART)
11.56.43.NYATLAQE.  MNF(R=2,E=1)
11.56.44.AOYI0241.  &REL,200000.
11.56.45.AEOY0341.  &RETURN,LGO.
11.56.47.ACLY050T.  &CATLIST.
11.56.49.NYATLAQE.    CTIME =     2758MS  PSR LEVEL 1
11.56.49.NYATLAQE.    CORE NEEDED IS     0565418
11.56.51.ADEY030T.  &LPF,LAD.
11.56.52.NYATLAQE.  STOP
11.56.55.AEOY0341.  &REWIND.INQIO.
11.56.56.NYAASBGE.  UPDATE(F)
11.56.58.ATLQ0711.  &CATLIST,LO=F.
11.57.07.AEOY034T.  COMPASS,1=INQIO,L=0.
11.57.09.AEOY0341.   ASSEMBLY COMPLETE.
11.57.10.NYATHAPE.   TIME LIMIT.
11.57.11.NYATHAPE.  FATAL ERROR 130.
11.57.11.NYATNAQE.  JOB(UMEEE15,J1)
11.57.12.NYAASBGE.  READING INPUT
11.57.13.NYATNAQE.  PASSWOR
11.57.14.NYATNAQE.  MNF(O,E=3,K=O,L=O)
11.57.17.AAAA002M.  MSORT,L024.
11.57.18.A1CY0651.  &LPF,OPULST.
11.57.19.AOYI0241.  &SUBMIT,JOBY,B.
11.57.22.NYATNAQE.    CTIME =     1329MS  PSR LEVEL 1
11.57.22.NYATNAQE.    CORE NEEDED IS     0551548
11.57.24.AZWQ040T.  REL.200000.
11.57.25.NYATOOIB.  JOB(UMEEE11,J4,MT1)
11.57.26.NYATOBMB.  PASSWOR
```

Figure 5.3: Example of System Dayfile

(e) System messages, such as initial dead start messages, recovery dead start messages, and Telex recovery messages.

An example of the System Dayfile is given in figure 5.3.

5.3 Dayfile Processing

5.3.1 Introduction

The Dayfile processing programs are a suite of programs developed for the purpose of:

(a) Data reduction of the Account and System Dayfiles.

(b) Analysis of the Dayfile data.

(c) Preparing data for input to the models of the IC Kronos system.

The Dayfile programs were developed over an extended period of time by the author in conjunction with two M.Sc. students, who developed some of the programs as part of their projects. An overall diagram showing the relationship between the programs and files used is shown in figure 5.4.

This section concentrates on the processing of the Dayfile data for input to the models of the Kronos system. The program referred to in this section take the raw Account and System Dayfiles as input, and generate two files, the B and J files. These files are input to pre-processor programs for the models of the system.

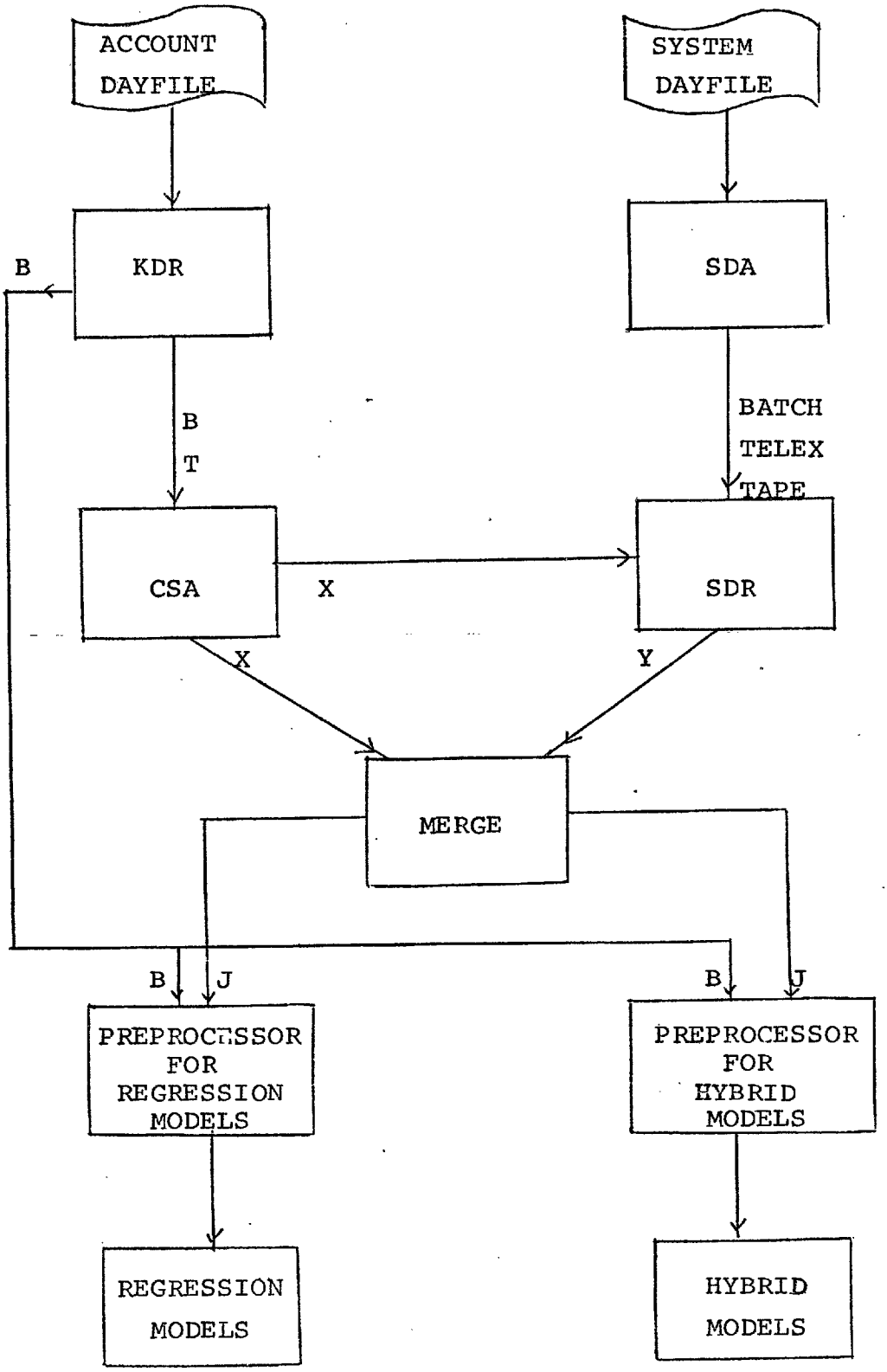


Figure 5.4: Dayfile Processing - Program and File Relationships

5.3.2 Account Dayfile Processing

5.3.2.1 Account Dayfile Reduction

The data reduction of the raw Account Dayfile into a more useable form is carried out by a program called Kronos Dayfile Reduction (KDR). KDR takes the raw Account Dayfile as input and generates two files, a Batch file and a Terminal file.

(a) The Batch (B) File

This file contains one record for each batch job processed and is ordered by job termination.

(b) The Terminal (T) File

This file contains one record for each terminal user and is ordered by user logout time.

A description of the data stored in the B and T files is given next. A full description of KDR, which was developed by J.L. Thompson is given in reference (T2).

The B File

The data collected for each batch job in the B file is as follows:

(a) Time the job was read through the card reader.
(b) The number of cards read.
(c) Time (in seconds) the job was scheduled for execution.
(d) Job name.
(e) Job category.
(f) Maximum Central Memory requested (as given on JOB card).
(g) Number of magnetic tape drives requested.

(h) Number of permanent file accesses made.
(i) Time the job terminated, in seconds.
(j) CPU time used, in milliseconds.
(k) Central Memory utilisation in word-hours.
(l) magnetic tape physical records input or output.
(m) Disc physical records input or output.
(n) Time the job's output was printed.
(o) Number of lines printed.
(p) State field. Indicator whether the record was accepted or rejected by the program. A record is rejected if:

(i) a message was missing
(ii) messages were out of sequence
(iii) an item of data appeared to be in error.

The T File

The data collected for each terminal user in the T file is as follows:

(a) Time user logged in, in seconds
(b) User's job name
(c) Port number user was logged in at
(d) Time user logged out, in seconds
(e) CPU time used, in milliseconds
(f) CM utilisation in word-hours
(g) Disc physical records input or output
(h) Number of characters output by system
(i) Number of characters input by system
(j) State field. Indicator whether the record was accepted or rejected.

5.3.2.2 Analysis of the System Load

The program CSA was developed to determine the average batch and terminal load experienced by each batch job during its execution phase. The average batch load calculated is the average number of batch jobs executing

concurrently with a given job. The average terminal load calculated is the average number of terminals logged in while a given job was in execution.

The program also calculates the average batch and terminal loads over fixed time intervals (e.g. 5 or 15 minutes). The average batch and terminal loads over a whole session are also calculated.

As no record of job rollin or rollout is maintained by the Dayfiles, the average batch load is a measure of all jobs in the execution phase competing for system resources. They may either be resident in Central Memory (CM) or rolled out to secondary storage.

CSA uses the B and T files prepared by KDR as input. CSA makes use of the job start and termination times from the B file, and the terminal login and logout times from the T file. CSA outputs the X file which has one record for each batch job containing the average batch and terminal loads during the job's execution.

CSA was developed by P.G. Jones and is described fully in reference (J1).

5.3.3 System Dayfile Processing

The System Dayfile reduction program, SDA, takes as input the raw System Dayfile and outputs three files.

(a) BATCH

This file has one record for each batch control card (equivalent to a job step) processed by the system. The initiation time of the job step and the jobname of the job it originated from are recorded in the file.

(b) TELEX

This file has one record for each terminal command (executed as a job step at a control point) processed. The initiation time of the command and the job name of the terminal user session it originated from are recorded in the file.

(c) TAPE

All System Dayfile messages relevant to magnetic tape processing are recorded in this file.

A second analysis program, SDR, uses the three files prepared by SDA and generates the Y file which contains one record for each batch job containing:

(a) The number of control cards (job steps) in each batch job processed.

(b) The number of batch job steps initiated by the system for batch jobs executing concurrently, with a given batch job. This provides a measure of batch activity.

(c) The number of terminal commands whose execution was initiated during a given job's time in the execution phase. This gives a measure of terminal activity.

A third program, MERGE, merges the X file generated by CSA and the Y file generated by SDR into one file called the J file. The J file also contains one record for each batch job and is ordered by job start time.

In addition to items (a) - (c) from the Y file, the J file also contains the following items from the X file:

(d) The average batch load experienced by a job during execution.

(e) The average terminal load experienced by a job during execution.

The three programs, SDA, SDR and MERGE were developed by P.G. Jones and are described fully in reference (J1).

5.4 The Dayfile Data Collected

The Dayfile data used for the performance evaluation and modelling of the IC system, was collected during three separate periods between July 1973 and April 1975.

(i) The first period was in July 1973. The morning and afternoon sessions of the 16th and 18th July were monitored on the CDC 6400 running the Kronos 2.0 operating system. Only the contents of the Account Dayfile were collected. KDR was used to process the Account Dayfile. Another program was written to extract measures of terminal loading averaged over fifteen minute intervals. No measures of batch loading were used in the analysis of this period.

(ii) The second period was in the spring of 1974. Four morning sessions (20th May, 12th, 17th and 18th June) and two afternoon sessions (17th and 18th June) were monitored on the CDC 6400 running the Kronos 2.0 operating system. The contents of both the Account and System Dayfiles were collected. All the programs described in this chapter were used for the Dayfile reduction and analysis.

(iii) The third period covered the first part of 1975. Two sessions in January 1975 (the morning of the 27th and the afternoon of the 30th) and two sessions in April 1975

(mornings of the 28th and 30th) were monitored on the CYBER 73 running the Kronos 2.1 operating system. The contents of the Account and System Dayfiles were collected and analysed.

5.5 Limitations of the Kronos Dayfile for Performance Evaluation and Modelling

The CDC Dayfile is primarily an accounting tool, and consequently the data gathered is oriented in that direction. The data collected is primarily at the macro (job) level. Analysis of this data can provide much valuable information about the characteristics of the workload and on overall throughput rates. In particular, it can provide:

(i) A detailed workload profile of the system. For example, the distribution of jobs by their resource (CPU, Memory, I/O) utilisation or compiler usage.

(ii) System throughput rates. For example, the mean and standard deviations of elapsed time for different classes of jobs at different times of the day/month/year.

(iii) Information on peak demands and loads on the system. For example, the number of terminal users logged in, length of job input and output queues, the average CPU utilisation over a whole session and its distribution between, for example, the batch and timesharing workloads.

Examples of Dayfile analysis are given in (G5), (G6), (L4) and (S10).

Accounting data has been used in the past in computer system modelling. The CDC Dayfile was used as the source of workload and performance data in simulating a CDC batch computer system (N3). The accounting subsystem on IBM OS/360 systems, the System Management Facility (SMF) (I3) was used as the source of data for simulating OS/360 systems (W2).

There are, however, many disadvantages to using the Kronos Dayfile as the only source of workload and performance data for the evaluation and modelling of the IC Kronos system. This is in part due to the nature of the Dayfile, but also because the Kronos Dayfile is more limited, in the amount and type of data collected, than the Dayfiles of other CDC computer systems, e.g. the CERN SCOPE computer system (L4). It is also considerably more limited than the IBM SMF package, which collects data at the job step level (I3).

The principle limitations of the Kronos Dayfile are:

(a) Data is collected at the job level for batch jobs and the session level for terminal users. For batch jobs the job start and end times are recorded; and the jobs resource utilisation is recorded at job termination. For a terminal user, login and logout times are recorded and the user's resource utilisation is recorded at logout time.

Consequently for both batch jobs and terminal users, no indication is given of the distribution of resource utilisation over the job or session's lifetime. This makes it difficult to effectively analyse the performance of the batch workload. It makes it virtually impossible to analyse the perfor-

mance of the timesharing workload. This is because a terminal user can spend large portions of the session thinking or doing trivial (in their demand on the system) operations such as typing in a program, followed by bursts of activity such as compilations and executions.

(b) Although the initiation of each batch job step and terminal command is recorded in the System Dayfile, no indication is given of the duration of the job step, nor of the resources used by the job step. Consequently no serious attempt can be made at analysing performance at the job step level. Similarly, the system can only be modelled at the job and not the job step level.

(c) No indication whatsoever is given of rollin/rollout activity. Thus, although a job's start and end times may be determined from the Dayfile, there is no means of finding out if and when a job was rolled out, for how long and for what reasons. In a dynamic system like the Kronos system, this is a severe limitation.

(d) The available measure of CM utilisation is crude. At job termination, a space/time measure is recorded in the Dayfile. The figure is computed for each period when the CM allocated to the job is constant by multiplying the memory allocated with the CPU time used. This figure is summed for the whole job and converted to Kiloword-hour units.

Dividing the Kiloword-hour utilisation by the CPU utilisation gives an estimate of the average memory used by the job during execution. This measure is in many cases not accurate because:

(i) Kiloword-hours is not a suitable unit for small jobs: if the CPU time is small, then the value of Kiloword-hours recorded (and hence the estimated average CM) is zero. Units of word-hours or Kiloword-seconds would be more suitable.

(ii) If the CPU utilisation varies widely at different stages of the job, then the estimate of average CM is biased towards memory sizes for which large CPU times were recorded.

(e) The I/O measures are crude. The total number of physical records transferred, which is the only value of I/O activity recorded, gives no indication of how many I/O requests were made, nor how long the requests took to be serviced. In addition, no indication is given of the channels used, channel time consumed, nor of peripheral processor activity.

(f) The time at which system jobs commence execution is not recorded in the Dayfile. Systems jobs have priority over all other jobs for the central processor, consequently the execution of system jobs can delay other jobs.

(g) No indication is given of overall system activity, for example when the system is heavily utilised, CPU bound or I/O bound.

5.6 Conclusions

The main effects of these limitations on the performance evaluation and modelling of the Kronos system are:

(a) Timesharing Subsystem

The data available is insufficient to allow an effective performance evaluation of the time sharing subsystem. Neither is the data sufficient to allow the timesharing subsystem to be modelled.

(b) Batch subsystem

The data is sufficient to allow an attempt to be made at a high level performance evaluation and modelling of the batch subsystem. However, this can only be carried out at the job and not at the job step level.

CHAPTER 6: REGRESSION MODELLING OF THE IMPERIAL COLLEGE SYSTEM

6.1 Introduction

This chapter describes the first attempt at the regression modelling of the Imperial College system. The implementation aspects of the regression analysis are described in section 6.2. An introduction to the regression models is given in 6.3. The models of the short and long job workloads are described in 6.4 and 6.5 respectively. The models are assessed in section 6.6. Appendix A presents an overview of multiple linear regression analysis and then describes the Forward Selection Regression procedure which was used in this analysis.

6.2 Implementation Aspects of Regression Modelling

6.2.1 Introduction

The data used for constructing the models was derived from the Kronos Account and System Dayfiles as described in Chapter 5. The B and J files are input to a preprocessor program which merges and sorts the data into the appropriate form for the Forward Selection Regression Program. The framework for the program was written by the author. The program uses a number of subroutines from the IBM Scientific Subroutine package (I1), which has been converted to run on CDC 6000 systems.

6.2.2 Preprocessing

The Dayfile processing programs take as input the Kronos Account and System Dayfiles for a particular session and output two files, the B and J files. The B file, which is ordered by job termination, consists of a job summary record for each job processed in the session. Each summary record holds measures of the resources demanded by the job

during its execution. The J file, ordered by job commencement, contains various measures of the load on the system during each job's lifetime.

The B and J files are input to the Preprocessor which merges and sorts the two files. The Preprocessor will output a number of different files depending on the input parameters specified. Each file consists of a set of observations, where each observation (one per job) consists of a set of job characteristics. These are listed in section 6.3.2.

The different output files are:

(i) The_All_Jobs_File which contains an entry for every job processed in the session.

(ii) The_Short_Job_File which contains an entry for every short job (J1 category) processed.

(iii) The_Long_Job_File which contains an entry for every long job (non-J1 category) processed.

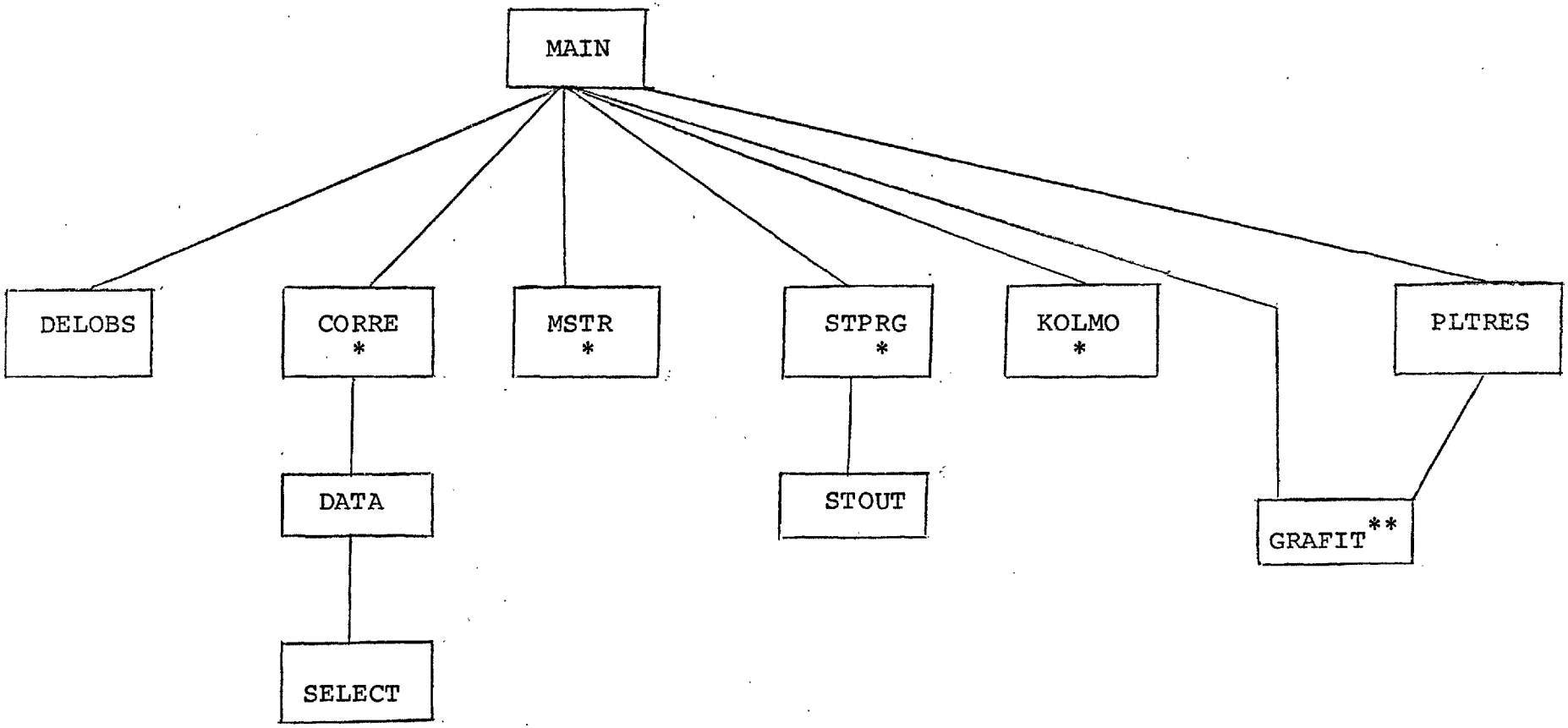
(iv) The_Long_Tape_Job_File which contains an entry for every long job processed that used magnetic tapes.

(v) The_Long_Non-Tape_Job_File which contains an entry for every long job processed that did not use magnetic tapes.

These files are in the appropriate format for input to the Forward Selection Regression Program.

6.2.3 The_Regression_Program

A block diagram showing the overall structure of the Forward Selection Regression Program is shown in figure 6.1. A short description of each routine follows:



* IBM Scientific Subroutine Package

** I.C. Program Library

Figure 6.1: Block Diagram of Forward Selection Regression Program

MAIN is the main routine. Its main function is to coordinate the execution of the other routines.

DELOBS is called by MAIN if any observations are to be excluded from the run. DELOBS sets up a vector of observations to be excluded.

CORRE is called by MAIN to calculate the means, standard deviations and simple correlations coefficients of all the variables.

DATA is called by CORRE to read the file of workload data prepared by the Preprocessor. The file contains the set of observations (one observation corresponds to a job) which are to be fitted. DATA makes use of the deletion vector set up by DELOBS to exclude unwanted observations for this run.

SELECT is called by DATA to prepare any second order independent variables as required e.g. CPU^2 or CPU*DPRU terms. (see section 6.3.2).

MSTR is called by MAIN to carry out some matrix manipulation prior to calling STPRG.

STPRG performs the forward selection regression procedure as described in Appendix A. The criterion for stopping the selection is whether R^2 has been improved by a specified amount, which is an input parameter to the program.

STOUT is called by STPRG to output the results of each step of the forward selection procedure.

KOLMO is called by MAIN to perform a Kolmogorov-Smirnov one-sample test on the residuals (S8). This tests the hypothesis that the residuals are normally distributed (Appendix A).

GRAFIT is a plotting routine. It is called by MAIN to plot the residuals against the estimated and observed values of the dependent variable.

PLTRES is called by MAIN if further plots are required. The residuals and the dependent variable are plotted against specified independent variables. GRAFIT is called to do the plotting.

CORRE, MSTR, STPRG and KOLMO are subroutines in the IBM Scientific Subroutine Package. GRAFIT is a subroutine in the Imperial College ICLIB program library.

6.3 Early Regression Models of the Imperial College System

6.3.1 Introduction

The initial attempt at the modelling of the Kronos System involved building regression models of the batch workload.

For the reasons given in Chapter 3, the job elapsed time was used as the dependent variable. The independent variables were of three types:

(a) Variables representing a job's resource demands.

(b) Variables representing measures of the overall load on the system. In this initial analysis, no variables relating to the batch load were available.

(c) A variable identifying the job class of the job.

Initially, models were constructed of the total batch workload. Models were then constructed of classes of jobs in the workload. The classes described here are the short (J1) and the long (non-J1) classes of batch jobs. The class of long jobs was then classified further into jobs which used magnetic tapes and jobs that did not.

The workload data used for building the models was gathered over four sessions on two separate days, the 16th and 18th July 1973. On each day there was a morning and afternoon session, separated by a system session. All the values of the dependent and independent variables used in building the model were extracted from the Account Dayfile using the program KDR (see 5.3.2.1). The System Dayfile was not used at all in this initial analysis.

6.3.2 The Independent Variables

In the construction of every model, a set of independent variables were available for selection by the Forward Selection Regression Program. The criterion for stopping the selection was when the inclusion of a new variable improved R^2 by less than a specified amount, e.g. 1%.

The independent variables were:

a) Job's resource demands:

- (i) Maximum CM requested (MAXCM)
- (ii) Average CM used (AVCM)
- (iii) Number of magnetic tapes required (NMT)
- (iv) CPU time (CPU)
- (v) Word hours (product of CPU time x CM used) (WH)
- (vi) Disc physical records input or output (DRRU)
- (vii) Magnetic tape physical records input or output (MTPRU)
- (viii) Number of permanent file requests (NPFREQ)

b) Measure of interactive load (averaged on a 15 minute basis):

- (i) Average terminal load while job was in execution (AVT) (average number of terminals logged in).

(ii) An approximate estimate of the interactive CPU demand per terminal hour (CPUSPH)

c) Class of job within workload

(i) Job Category (JCL)

Initially models were constructed for each of the four samples. In cases where different samples were merged, two dummy independent variables were used whose coefficients,if significant, would indicate a dependency on the day or time of day when the jobs were run.

6.3.3 Analysis of the Workload

The main features of the batch workload for the four sessions are displayed in table 6.1. The short job workload (J1 jobs) accounted for over 80% of the jobs processed in all four sessions. However the CPU utilisation of the short job workload was much less, varying from 24% in the 16/7 a.m. sample to 56% in 18/7 p.m. sample. The mean CPU time for short jobs varied from 4.6 seconds to 5.6 seconds. The mean CPU time for long jobs was much larger, varying from 26 seconds, on 18/7 p.m. to 108 seconds on 16/7 a.m.

It is thus clear even from this superficial study of the workload that the short jobs though much larger in number, in general consume considerably less resources than the large jobs, which are few in number. These characteristics are fairly typical of university computing environments.

The characteristics of the short and long job workloads are shown in tables 6.2 and 6.3 respectively.

Table 6.1: Characteristics of the 1973 Batch Workload

Class	Characteristic	16/7/73 a.m.	18/7/73 a.m.	16/7/73 p.m.	18/7/73 p.m.
All Jobs	number of jobs	432	487	723	811
	Total CPU time (seconds)	7179	4390	8685	6950
	Mean CPU time (seconds)	16.6	9.0	12.0	8.6
Short Jobs	number of jobs	378	433	594	692
	percentage of all jobs	87.5	89.0	82.0	85.3
	Total CPU time (seconds)	1739	1990	3085	3880
	Mean CPU time (seconds)	4.6	4.6	5.2	5.6
	percentage of batch CPU time (seconds)	24.2	45.5	35.8	55.8
	Mean elapsed time (seconds)	23.8	34.8	32.0	43.0
Long Jobs	number of jobs	54	54	129	119
	percentage of all jobs	12.5	11.0	18.0	14.7
	Total CPU time (seconds)	5440	2400	5600	3070
	Mean CPU time (seconds)	108.0	44.8	43.5	25.8
	percentage of batch CPU time	75.7	54.6	64.5	44.2
	Mean elapsed time (seconds)	865	590	767	396

Table 6.2: Characteristics of the 1973 Short Job Workload (Four Samples)

	16/7 a.m.	18/7 a.m.	16/7 p.m.	18/7 p.m.
Elapsed time(secs) m.	23.8	34.5	32.0	43.0
s.d.	49.5	42.0	55.4	51.7
CPU time (secs) m.	4.6	4.6	5.2	5.6
s.d.	5.3	5.2	5.2	5.2
Word hours m.	21.5	22.1	25.9	27.2
s.d.	22.6	24.3	25.3	25.9
disc records transferred (p.r.u.) m.	243.9	262.4	324.0	314.1
s.d.	286.2	366.0	342.4	469.0
terminal load m.	16.6	23.9	19.0	19.2
s.d.	3.2	7.2	5.3	8.7
Number of permanent file requests m.	2.7	2.2	2.4	2.4
s.d.	6.3	1.5	1.5	1.8
Number of short jobs	378	433	594	692

Key
m: mean
s.d.: standard deviation
p.r.u.: physical record units

Table 6.3: Characteristics of the 1973 Long Job Workload (Four Samples)

	16/7 a.m.	18/7 a.m.	16/7 p.m.	18/7 p.m.
Elapsed time(secs)m.	865	590	767	396
s.d.	1595	1122	1771	1131
CPU time (secs)m.	108.0	44.5	43.5	25.8
s.d.	254.0	128.0	127.0	65.4
Word hours m.	586	295	242	133
s.d.	1582	922	740	376
Disc records (p.r.u)m	2723	1223	1663	907
s.d.	8973	2398	4515	1635
Tape requests m.	0.4	0.4	0.4	0.2
s.d.	0.5	0.6	0.5	0.5
Terminal load m.	17.5	20.0	15.4	19.2
s.d.	3.6	11.3	7.2	6.5
Number of long jobs	54	54	129	119
Number of tape jobs	20	17	54	24
Number of non-tape jobs	34	35	75	95
Number of J4 jobs	47	47	93	107
Number of J7 jobs	7	7	36	12

Key: m: mean
s.d.: standard deviation
p.r.u.: physical record units

6.3.4 Regression Models of the Total Workload

Regression models of the total workload were constructed for each of the four samples. These are displayed in figure 6.2. In each equation, the order of the independent variables, from left to right, represents the order in which these variables were selected by the Forward Selection Regression procedure.

In three of the four models (16/7 a.m., 18/7 a.m. and 18/7 p.m.) over 80% of the variation (R^2) was explained by the model, and in each case WH made the most significant contribution. The values of the regression coefficients of WH, however, vary considerably. In the fourth model (16/7 p.m.), the fit is much poorer, and the variable making the most significant contribution is DPRU. This suggests that some heavily I/O bound jobs were dominating this sample. In none of the samples, did the independent variables representing terminal load make a significant contribution.

Analysis of the residuals revealed that most of the residuals were negative and comparatively small in magnitude. However, a smaller number were positive and much larger in magnitude. In particular, the large positive residuals occurred for jobs with large elapsed times. Furthermore, the estimates for the dependent variable, namely job elapsed time, were negative for a large number of short jobs.

Thus, in spite of the large R^2, the models appeared unsatisfactory for a number of reasons. Firstly, the models were inconsistent, showing large differences in the variables selected and in the values of the regression coefficients. Secondly, the large number of negative estimates meant that it was unrealistic to expect this type of model of the total workload to be a satisfactory

Figure 6.2: Regression Models of the Total Workload

16/7/73 a.m. sample - 432 observations

$$Y = -316 + 0.8\ WH + 310\ JCL + 0.02\ DPRU$$
$$F = 1202 \quad s = 205 \quad R^2 = 0.89$$

16/7/73 p.m. sample - 723 observations

$$Y = -2808 + 0.2\ DPRU + 0.11\ MAXCM + 0.63WH + 388\ NMTS$$
$$F = 222 \quad s = 535 \quad R^2 = 0.55$$

18/7/73 a.m. sample - 487 observations

$$Y = -1052 + 2.9\ WH + 285NMTS - 15.4\ CPUT + 0.04\ MAXCM + 0.42\ MTPRU + 133\ JCL$$
$$R^2 = 0.81$$

18/7/73 p.m. sample - 811 observations

$$Y = -54 + 2.3WH + 0.11\ DPRU + 0.63\ MTPRU$$
$$F = 1257 \quad s = 190 \quad R^2 = 0.82$$

---

key: Y : Job Elapsed time (dependent variable)
Independent variables : Key in section 6.3.2
R^2 : proportion of variation explained by model
s : standard error of the residuals
F : F statistic

means of modelling short jobs. Thirdly, the large positive residuals meant that, for some jobs at least, a large amount of the variation was unexplained. It was therefore decided, in the first instance, to classify the workload into long and short job workloads and build regression models of each.

6.4 Regression Models of the Short Job Workload

6.4.1 Models of Individual Sessions

The short job workload consists of all jobs in the J1 category. The characteristics of the short job workload for the four samples is shown in table 6.2. Regression models were constructed for each of the four samples and are shown in figure 6.3. A regression model was also constructed using data pooled from the two morning samples, and is shown in figure 6.4.

Figure 6.3 shows that the amount of variation explained by the models, R^2, varies from as little as 9% on 16/7 p.m. to 50% on 16/7 a.m. The models were considered unsatisfactory because of this large unexplained variation. Furthermore, the models are substantially different, with different variables and different values of regression coefficients.

6.4.2 Analysis of Residuals

An analysis of residuals shows that the plots of residuals against the job elapsed time (the dependent variable), for the different models, all display similar characteristics. As an example, consider the residual plot against predicted job elapsed time for the 18/7 a.m. model, which is shown in figure 6.5. This shows that

Figure 6.3: Regression Models of the Short Job Workload (Individual Samples)

16/7/73 a.m. sample - 378 observations

$$Y = 289 + 4.3\ NPFREQ + 6.5\ CPUT - 0.01\ MAXCM - 0.8\ WH$$
$$F = 94 \quad s = 35.1 \quad R^2 = 0.50$$

16/7/73 p.m. sample - 594 observations

$$Y = -15 + 0.6\ WH + 1.7\ CPUSPH$$
$$F = 53.0 \quad s = 28.3 \quad R^2 = 0.09$$

18/7/73 a.m. sample - 433 observations

$$Y = 13.7 + 0.05\ DPRU + 0.39\ WH$$
$$F = 81.6 \quad s = 36 \quad R^2 = 0.28$$

18/7/73 p.m. sample - 811 observations

$$Y = 4 + 0.3\ DPRU + 2.9\ CPUT + 5.6\ NPFREQ$$
$$F = 92 \quad s = 43.8 \quad R^2 = 0.29$$

Figure 6.4: Regression Models of the Short Job Workload (Combined Morning Samples)

No. of observations = 811

$$Y = -5.2 + 4.2\ NPFREQ + 5.1\ CPUT + 0.03\ DPRU + 12.5\ DAY - 0.6\ WH$$
$$R^2 = 0.37$$

For key refer to figure 6.2

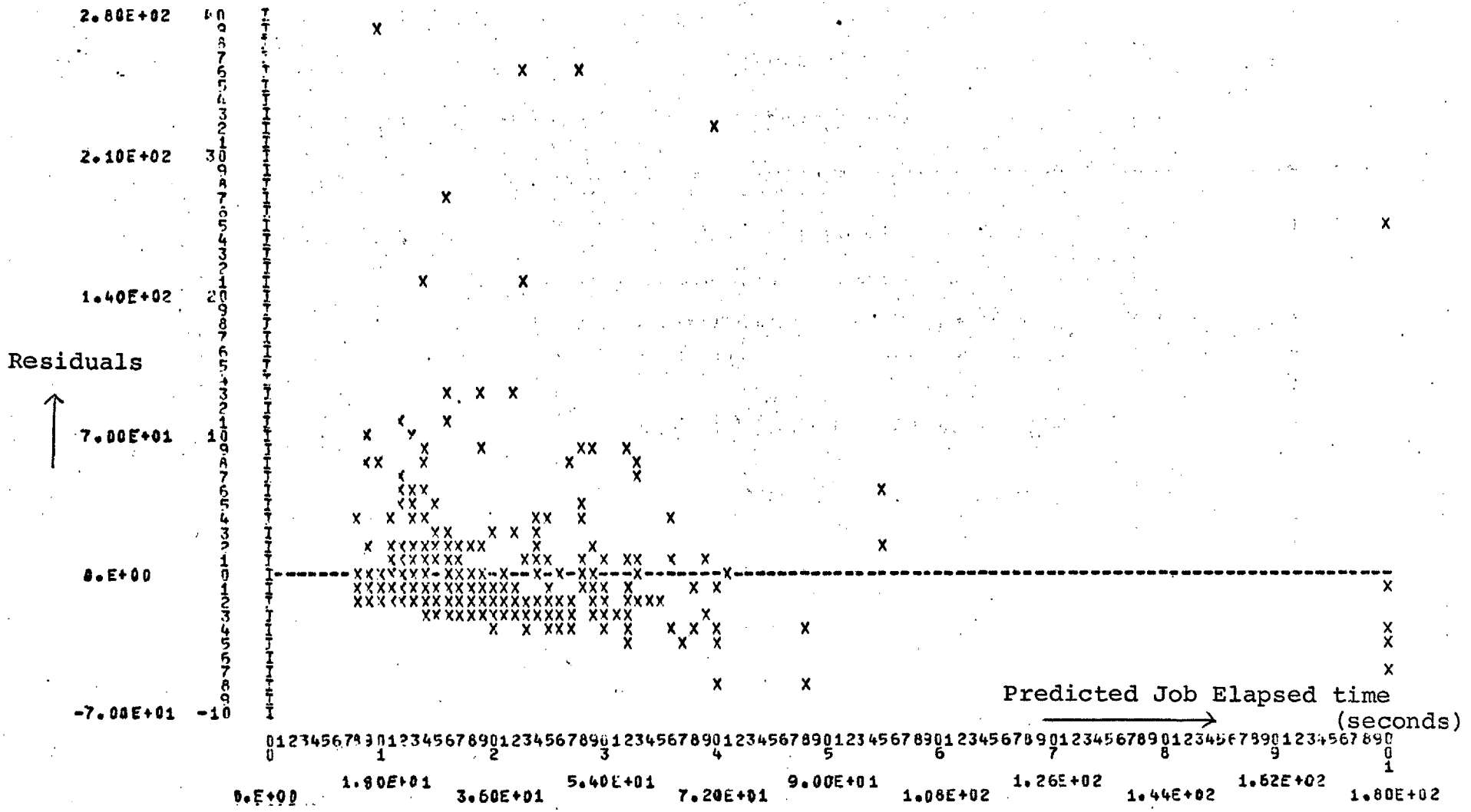


Figure 6.5: Plot of Residuals v Predicted Job Elapsed Time - Short Job Model (18/7 a.m. sample)

although all the negative residuals are comparatively small in magnitude, some of the positive residuals are much larger. The residual plots against the independent variables do not explain this variation. The plot of residuals against actual job elapsed time, however, (figure 6.6) reveals another interesting fact. In general, the negative residuals all occur for small values of job elapsed time, whereas the large residuals all occur at large values of job elapsed time. Furthermore, there is a tendency for the residuals to be positively correlated with actual elapsed time.

The residual plot suggests that there is some,possibly time-dependent,independent variable(s) which so far has not been taken into account in constructing the models. A possible explanation is the effect of the interactive workload on the batch workload. When the interactive load is high, batch jobs are rolled out of CM (because of their lower priority) and are not rolled in again till the interactive load subsides. The available measures of interactive load are averaged over fifteen minute intervals and have not been selected for inclusion in the model. Since the mean elapsed time for a short job is around 30 seconds, it would seem that a more suitable measure of interactive load should be at this level of resolution.

6.4.3 Comparison of Morning Sessions

A regression model was also constructed using workload data from the two morning samples (figure 6.4). A dummy independent variable was used to identify the day of the run. It was set to zero for all observations (one observation per job) in the 16/7 a.m. sample, and set to 1 for all observations in the 18/7 a.m. sample. If significant, the regression coefficient of the 'day' variable should be an estimate of the increased (or decreased) elapsed time caused by running the jobs on different mornings. The 'day' variable made a significant contribution to the model, suggesting that there was a significant difference in system behaviour on the two mornings.

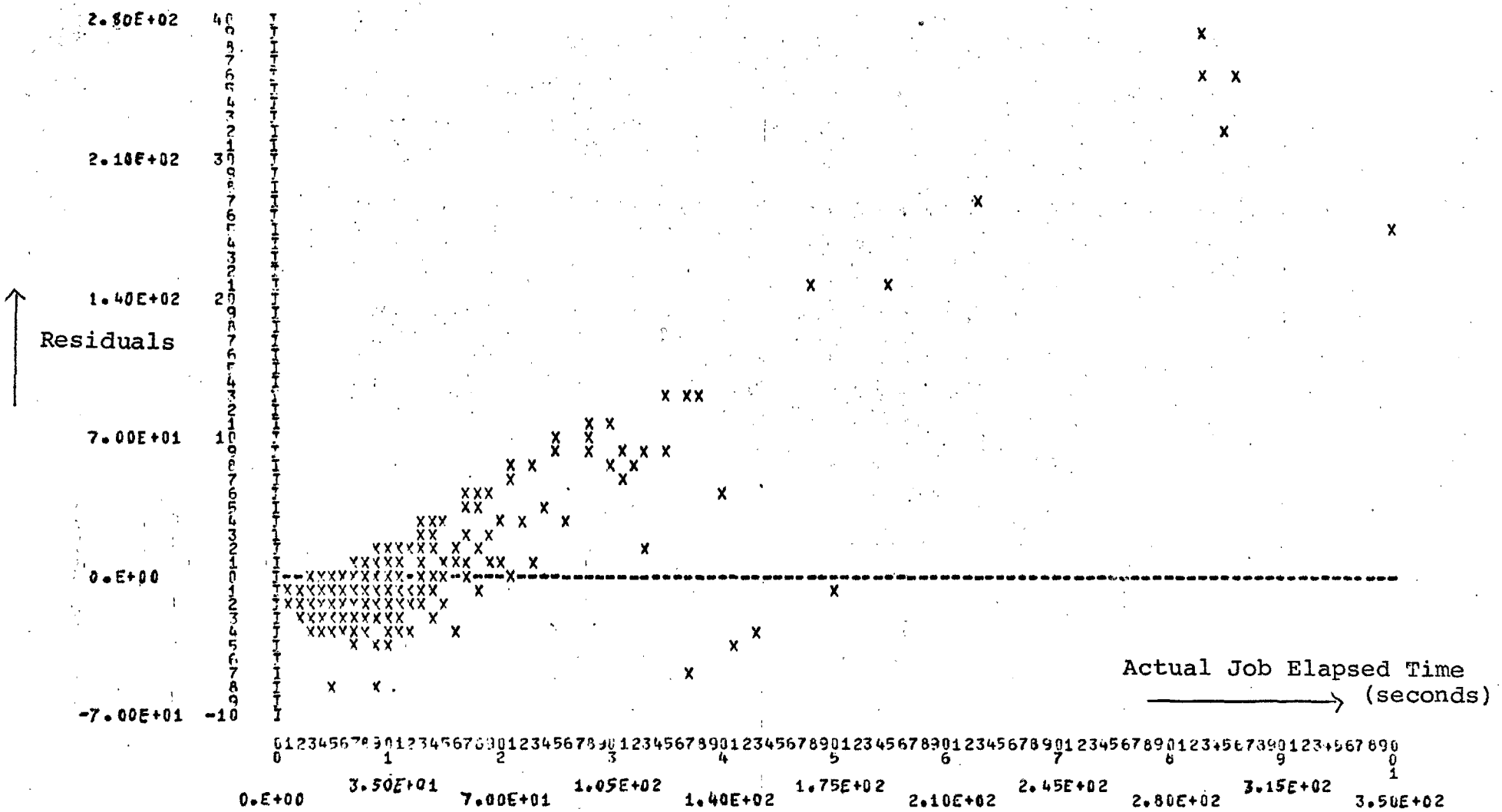


Figure 6.6: Plot of Residuals v Actual Job Elapsed Time - Short Job Model (18/7 a.m. sample)

Comparing the characteristics of the two morning samples (table 6.2) indicates that although the mean job 'CPU time', 'word hour', 'disc record transfers', and 'number of file requests' correspond quite closely, the mean job elapsed time in the first sample (23.8 seconds) was considerably shorter than in the second (34.5 seconds). This difference in short batch job performance, reflected in the difference of 10.7 seconds in the mean job elapsed times, corresponds fairly closely to the regression coefficient of the 'day' variable, 12.5 seconds.

Although not recorded as having made a significant contribution to the variation, the terminal load was considerably higher in the second sample than in the first. A peak of 33 terminals as compared to 19 was recorded. If, as seems highly probable, the two measures of terminal load used in this analysis do not provide a good enough measure of interactive load to have a significant effect on the model, then it is likely that the difference in system behaviour is being attributed to the 'day' variable instead of the terminal load. This problem is analysed further in chapter 7.

6.5 Regression Models of the Long Job Workload

6.5.1 Models of Individual Sessions

Regression models were constructed for each of the four samples of long (i.e. non-J1) jobs (figure 6.7). Considerably more of the variation was explained in these models than in the short job models. In general, there was considerable similarity, for each sample, between the model constructed for the long job workload and the total workload (figure 6.2). This indicates that it is the long jobs, a small part of the workload in numbers, which determine the form of the model. This is because the contribution to the total sum of squares by the long jobs is much greater than the contribution by the short jobs, even though these are much more in number. Hence the contribution due to the long jobs dominates the model.

Figure 6.7: Regression Models of the Long Job Workload

16/7/73 a.m. sample - 54 observations

$$Y = 28 + 0.79\ WH + 676\ JCL + 0.024\ DPRU + 339\ NMTS$$
$$F = 112 \quad s = 590 \quad R^2 = 0.90$$

16/7/73 p.m. sample - 129 observations

$$Y = -3579 + 0.22\ DPRU + 0.15\ MAXCM + 0.54\ WH$$
$$F = 45 \quad s = 1245 \quad R^2 = 0.52$$

18/7/73 a.m. sample - 54 observations

$$Y = -3135 + 0.67\ WH + 0.13\ MAXCM + 423\ NMTS$$
$$F = 40.0 \quad s = 627 \quad R^2 = 0.75$$

18/7/73 p.m. sample - 119 observations

$$Y = -134 + 2.3\ WH + 0.19\ DPRU + 213\ NMTS$$
$$F = 209 \quad s = 450 \quad R^2 = 0.85$$

For key refer to figure 6.2

In three of the four models, more than 80% of the variation was explained and WH made the most significant contribution. One model (16/7 p.m.) displayed different characteristics. Only 52% of the variation was explained and DPRU made the most significant contribution. The analysis of the residuals for this model revealed some outliers, in particular one job with most unusual characteristics.

In all models, the residuals are large and the standard errors high. In addition, there is a considerable variation between the models. It was concluded that the models were therefore not suitable in this form.

The long job workload includes some jobs which use magnetic tapes and others which do not. There is likely to be a substantial difference in characteristics between these two classes of jobs. Hence, it was decided that a further classification of workload, into long jobs that used tapes and those that did not would be appropriate.

6.5.2 Models of the Non-Tape and Tape Classes of Long Jobs

Models were constructed for the long non-tape job class and the long tape job class of jobs. These were compared with each other and with the long jobs model for the same data. Consider the example in figure 6.8, which shows the three models. All three models were built from observations for the two morning samples. A 'day' dummy variable was used, but it did not make a significant contribution to any of the models.

Apart from the magnetic tape term in the long jobs model, the long job model and the non-tape long jobs models match quite closely. The regression coeffi-

Figure 6.8: Comparison of Long Job Models (Combined Morning Samples)

Long Jobs Model - 106 observations

$$Y = 1890 + 0.80\,WH + 0.08\,MAXCM + 0.03\,DPRU + 371\,NMTS$$
$$R^2 = 0.83$$

Long Non Tape Jobs Model - 69 observations

$$Y = -2106 + 0.78\,WH + 0.09\,MAXCM + 0.03\,DPRU$$
$$F = 178 \quad s = 546 \quad R^2 = 0.89$$

Long Tape Jobs Model - 37 observations

$$Y = 322 + 2.4\,WH + 615\,JCL$$
$$F = 36 \quad s = 477 \quad R^2 = 0.63$$

Key : refer to figure 6.2

cient of the 'number of magnetic tapes' variable represents an estimate of the time taken to mount a magnetic tape, 371 seconds.

However, the tape jobs model shows considerable differences. Firstly, the amount of variation explained by the model is much lower. This is almost certainly due to the fact that the time taken to load a magnetic tape is not available from the Account Dayfile, and consequently could not be included in the model.

Table 6.4 compares the characteristics of the tape and non-tape classes of long jobs for the combined morning samples. It can be seen that although the mean elapsed times for the two classes are of the same order, the CPU and word-hour utilisation figures are considerably smaller for the tape class. Thus the contribution to the total sum of squares by the long non-tape jobs appears to be dominating the models of the long job workload and indeed the models of the total workload.

6.5.3 Models of the Long Non-Tape Job Sample

Linear regression models of both the first order and second order variety were built for the long non-tape job samples for the 16th and 18th July respectively. A dummy 'time of day' variable was used to distinguish between observations in the morning sample and afternoon sample. An analysis of residuals was also carried out, leading to the exclusion of outliers, and the construction of more refined models.

Figure 6.9 shows the first order model, second order model, and second order model with outliers excluded, for the 16/7 and 18/7 samples respectively. In the second order model, second order independent variables of the quadratic (e.g. CPU^2, $DPRU^2$) and product (e.g. CPU*DPRU,DPRU*AVCM) type were made available for selection by the Forward Selection Regression procedure.

Table 6.4: Characteristics of Tape and Non-Tape Classes of Long Jobs [16th and 18th July 1973 morning Samples]

Characteristic		Tape Jobs	Non-tape Jobs
Number of jobs		37	69
Elapsed time (secs)	m.	731	1090
	s.d.	823	1952
CPU time (secs)	m.	21.5	163.2
	s.d.	27.0	308.0
Word hours	m.	122	894
	s.d.	199	1937
Disc records transferred (p.r.u.)	m.	1648	3407
	s.d.	4034	10841
Number of tape requests	m.	1.1	-
	s.d.	0.4	
Tape records transferred (p.r.u.)	m.	729	-
	s.d.	2079	

Key: m: mean
s.d.: standard deviation
p.r.u.: physical record units

Figure 6.9a - Long Non-Tape Job Models

16/7/73 sample - 109 observations

First order Linear Model

$$Y = 88 + 0.64\ WH + 0.08\ DPRU + 576\ JCL$$
$$R^2 = 0.57 \quad s = 1156$$

Second order Linear Model

$$Y = -95.6 + 0.89\ WH + 0.047\ DPRU*AVCM - 0.00034\ CPU*DPRU$$
$$R^2 = 0.70 \quad s = 996$$

Second order Linear Model with outliers excluded

$$Y = 9309 + 0.77\ WH + 0.00001\ MAXCM^2 - 0.698\ MAXCM + 0.00001\ DPRU^2$$
$$R^2 = 0.91 \quad s = 361$$

Figure 6.9b (continued)

18/7/73 sample - 130 jobs

First order Linear Model

$$Y = -89 + 1.14\ wH + 0.23\ DPRU$$
$$R^2 = 0.7 \quad s = 630$$

Second order Linear Model

$$Y = -41 - 32\ CPU - 0.03\ CPU^2 + 0.003\ CPU*DPRU + 0.0016\ CPU*MAXCM$$
$$R^2 = 0.90 \quad s = 364$$

Second order Linear Model with outlier excluded

$$Y = -66 - 30.8\ CPU - 0.03\ CPU^2 + 0.003\ CPU*DPRU + 0.0016\ CPU*MAXCM$$
$$R^2 = 0.96 \quad s = 236$$

For key refer to figure 6.2

Figure 6.9 shows that for both samples, the second order model is a noticeable improvement over the first order model, with higher R^2 and lower standard error. An analysis of residuals reveals which jobs are outliers. These are usually jobs which make uncharacteristic resource demands, e.g. very large I/O demands. Excluding these outliers results in a further improvement in the model. Furthermore the exclusion of outliers can result in a substantial change in the form of the model. This is noticeable in the 16/7 models shown in figure 6.9a. This is because the presence of outliers can distort the model.

Consider the 18/7 sample. With the first order model, there are three large outliers exceeding 2000. With the second order model, there is one large outlier. When this is excluded, a good fit is obtained, as shown in figure 6.10. However, it is clear that there are four large observations with large resource demands, and these have been fitted well. It is these four observations which account for the major part of the good fit, and of the final form of the model.

Comparison of this model with its counterpart for the 16/7 sample, shows that the two models are not consistent. Each model is dominated by a few jobs, and so each model is dependent on the characteristics of those few jobs.

6.5.4 The Domination Effect

The examples described in this chapter demonstrate that a small number of jobs with large resource demands can dominate a model constructed from a much larger sample of data.

An experiment was carried out to determine approximately how many jobs were responsible for this

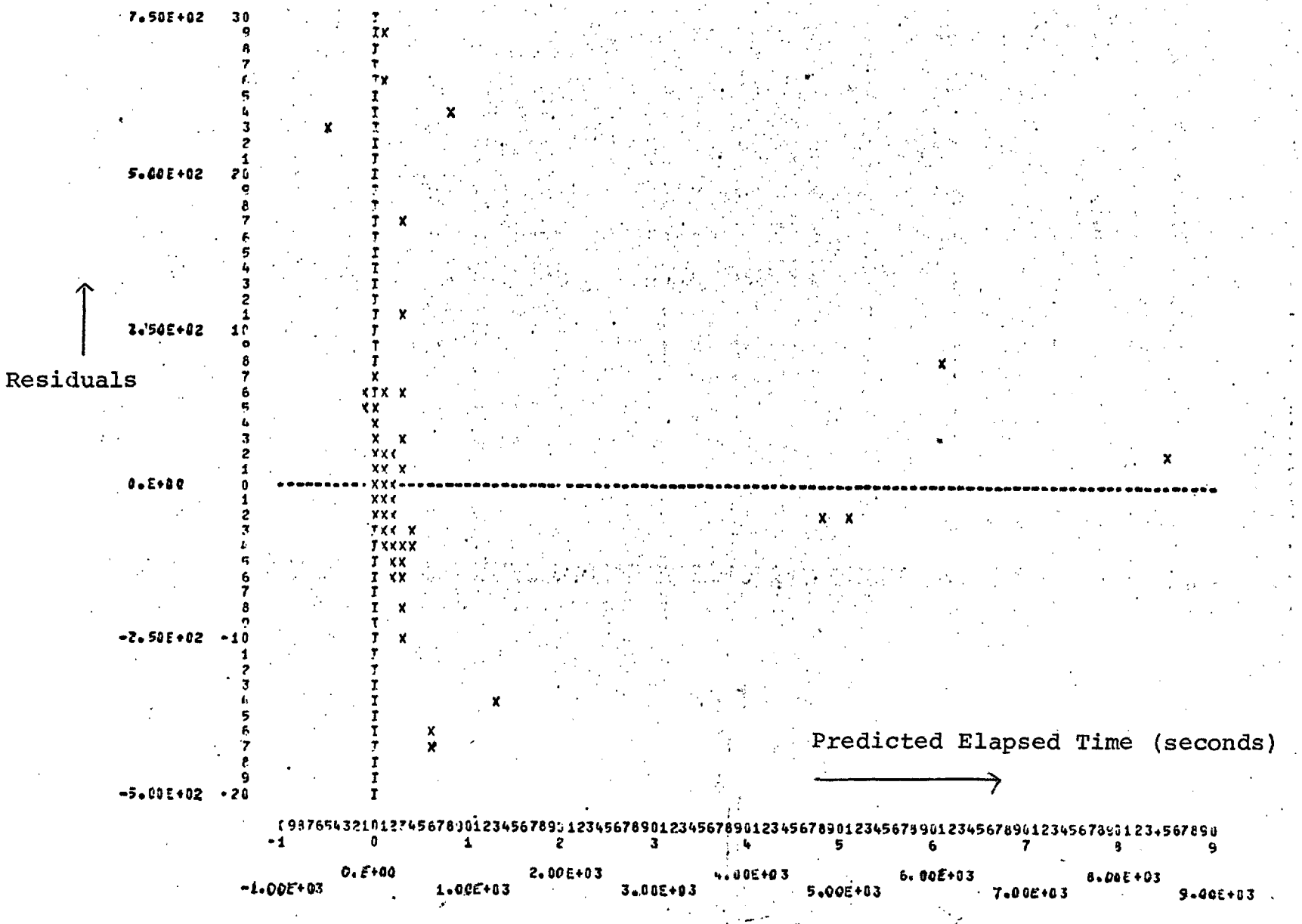


Figure 6.10: Residual Plot: Second Order Long Job Model with Outlier excluded (18/7 sample)

domination effect. Two models of the long non-tape job class were constructed, and are shown in figure 6.11. The first model was constructed for all the jobs in this class from all four samples of data. The second model used a subset of this data. All jobs which used more than 66 seconds CPU time, 23 in all, were excluded. Outliers were also excluded from both models. The first model has a large R^2 and is very similar in form to the second order model (with outliers excluded) constructed for the 16/7 sample (figure 6.9a). This implies that it is jobs from the 16/7 sample which dominate both models. However, when the 23 largest jobs are excluded, figure 6.11 shows that the form of the model changes radically. R^2 is reduced considerably from 0.92 to 0.40. It is, thus, the small number of jobs with the largest resource demands which dominate the long non-tape job model. If the 5 outliers excluded are also counted then the number of jobs which dominate the model is 28.

Since it was the jobs from the long non-tape job sample which dominated the long job workload and total workload models, it appears that the whole sample of 2453 jobs, is dominated by at most 28 jobs.

6.6 Conclusions

The main conclusions of this first attempt at the regression analysis of the Imperial College system are that the models built for long and short job workloads are unsatisfactory, but for different reasons.

(a) Short Job Workload

The models of the short job workload are unsatisfactory with low R^2 and inconsistent regression coefficients. An analysis of residuals reveals a regular trend in all the models, namely a large number of small negative residuals with a smaller

Figure 6.11 - The Domination Effect

Long Non-Tape Job Model - 231 observations

$$Y = 9020 + 0.72\ WH + 0.00001\ MAXCM^2 - 0.68\ MAXCM + 0.00001\ DPRU^2$$

$$R^2 = 0.92 \quad s = 378$$

Long Non-Tape Job Model (Jobs using under 66 seconds CPU time) - 208 observations

$$Y = -31 + 1.04\ WH + 2.3\ CPUSPH$$

$$R^2 = 0.40 \quad s = 111$$

number of much larger positive residuals. In general, the negative residuals occur for small values of actual job elapsed time, whereas the large residuals occur for large values of elapsed time. The positive residuals also appear to be positively correlated with elapsed time.

The presence of the large positive residuals is due to the actual elapsed time for some jobs being substantially larger in value than the predicted job elapsed time. This means that there is some additional, possibly time-dependent, factor contributing to the elapsed time, which is not taken into account by the models. This factor is likely to be associated with the fluctuating load on the system, resulting in batch jobs being rolled out of Central Memory for varying periods of time.

Insufficient measures were available in this analysis of the load imposed on the system by both the batch and interactive workloads. Only crude measures of the interactive load were used which were averaged on a 15 minute basis, compared with a mean job elapsed time of 30 seconds. No measures of the batch load were available at all. The influence of the system load on the short job workload is analysed further in Chapter 7.

(b) Long Job Workload

A large proportion of the variation in the models of the long job workload was explained. However, the regression coefficients were inconsistent and the standard error of the residuals large. It is the large non-tape jobs which dominate the model. An analysis of this subset of the workload showed that it was the very small number of jobs with the largest resource demands which dominate the models.

These jobs, 23 in all, form less than 1% of the total sample of 2453 jobs used in the analysis.

An important assumption in regression analysis is that the data used for constructing the models is representative (see Appendix A). To construct consistent models of the long job workload, it is necessary to have a much larger sample of long jobs. This means that the data for this analysis would need to be collected over a considerable period of time. The facilities for carrying this out were not available in the project. Moreover, to develop comprehensive models of the long job workload, workload data of a more detailed nature (e.g. at the job step level) would be a considerable advantage.

Consequently, it was decided not to proceed further with modelling the long job workload, but to concentrate instead on modelling the short job workload.

CHAPTER 7: THE WORKLOAD MODEL

7.1 Introduction

This chapter describes a further regression analysis of the Imperial College system at the workload level. Following the decision in Chapter 6, the short job (J1) workload was modelled extensively for six sessions in the spring of 1974 and for four sessions in the first four months of 1975. Whereas the Account Dayfile was the only source of data used for the analysis described in Chapter 6, the System Dayfile was also made available for this analysis.

Regression models are built which attempt to explain a short job's elapsed time (that is the real time from when a job is first scheduled for execution to the time it terminates) in terms of:

- the resources a job demands
- the load on the system, both batch and interactive.

The load measures are measures of the amount of competition for system resources that a job experiences. It was discovered in the previous analysis (Chapter 6) that the measures of system load used were insufficient. Further load measures were made available for this analysis, derived from both Account and System Dayfiles.

Sections 7.2 to 7.6 of this chapter, describe the regression analysis of the first set of data collected, in the spring of 1974. Sections 7.7 to 7.9 describe the regression analysis of the second set of data collected in the first four months of 1975.

In section 7.2, the characteristics of the workload in the spring of 1974 are presented and analysed.
In section 7.3, a first analysis aimed at modelling the Kronos system at the workload level is described, which used two morning sessions. The modelling exercise

showed that at certain periods of the day, the predicted elapsed times were considerably smaller than the actual elapsed times, resulting in large residuals. A further analysis of the Dayfiles is presented in section 7.4 which identified the causes of these delays. In section 7.5, a more comprehensive regression modelling exercise was carried out,using four morning sessions. In section 7.6, the analysis and modelling of two afternoon sessions is described.

The second set of data collected, in 1975, was on the CYBER system, which supported an entirely batch workload. Section 7.7 describes the models constructed of the batch workload in the absence of the timesharing load. Section 7.8 describes the validation of these models, which led to the construction of the Workload Model. Section 7.9 describes the modelling of a subset of the short job workload which did not experience any competition from other short jobs.

7.2 The Workload Data

7.2.1 The Sessions Analysed

The workload data used for the second analysis of the IC Kronos system,was gathered over six different sessions in the spring of 1974. During the period in question, a number of hardware faults were experienced. Over a dozen sessions were subjected to an initial Dayfile analysis. More than half the sessions were rejected, either because of unscheduled dead starts or due to failures in the Telex subsystem.

The sessions accepted were four morning sessions and two afternoon ones. The morning sessions were on 20th May, 12th, 17th and 18th June. The afternoon sessions were on the 17th and 18th June. Both the Account and System Dayfiles were collected for each session.

7.2.2 Characteristics of the Workload

The main characteristics of the workload are presented and discussed in this section. It was discovered during the analysis that the 17/6 morning session had a hangup for a period of about four minutes, when no messages appeared in either Dayfile. A magnetic tape job appeared to block while in possession of a channel. The blockage cleared when the job was dropped. This partly explains why the figures for batch and terminal CPU utilisation are on the low side for this session. Because of this, the 17/6 session has been ignored in the discussion in this subsection.

Table 7.1 displays the main characteristics of the batch workload for the four morning sessions. The number of jobs executed varied between 477 on 20/5 and 574 on 18/6. However, the highest batch CPU utilisation was recorded in the former session and the lowest in the latter. In all four sessions, over 80% of the jobs executed were short jobs. The CPU utilisation of the short job workload was in all cases around the 15% level. The long job workload in all cases used more CPU time than the short job workload, in one case more than twice as much. Most of the long job CPU time was accounted for by jobs which did not use magnetic tapes.

Table 7.2 displays the main characteristics of the terminal workload on the four sessions. It can be seen that the average terminal load over the whole session was substantially lower on 20/5, averaging 15, than it was on the other three mornings, when it averaged over 20. Figures 7.1 and 7.2 show how the terminal load varied over each of the 20/5 and 18/6 sessions respectively. The highest terminal CPU utilisation was recorded on 18/6. In reference G6, it was shown that a good measure of total system activity is given by the 'number of times no Peripheral Processor

Table 7.1: Characteristics of the 1974 Batch Workload

Class	Characteristic	20/5/74	12/6/74	17/6/74*	18/6/74
All Jobs	No. of jobs	477	524	532	574
	Total CPU time (secs)	6408	4978	4245	4751
	Mean CPU time (secs)	13.4	9.5	8.0	8.3
	CPU Utilisation	51%	41%	34%	39%
Short Jobs	No. of jobs	401(85%)	441(84%)	432(81%)	467(81%)
	Total CPU time (secs)	1888	1764	1945	1961
	Mean CPU time (secs)	4.7	4.0	4.5	4.2
	Mean elapsed time (secs)	23.7	40.2	32.2	32.3
	CPU Utilisation	15%	14%	15%	16%
Long Jobs	No. of jobs	76	83	100	107
	Total CPU time (secs)	4520	3214	2300	2790
	Mean CPU time (secs)	59.5	38.7	23.0	26.1
	Mean elapsed time (secs)	531	498	1017	614
	CPU Utilisation	36%	26%	19%	23%
Long Non-tape Jobs	No. of jobs	48	48	43	60
	Total CPU time (secs)	4119	2895	1798	2166
	Mean CPU time (secs)	85.7	60.3	41.8	36.1
	Mean elapsed time (secs)	712	666	414	679
	CPU Utilisation	33%	24%	14%	18%
Long Tape Jobs	No. of jobs	28	35	57	47
	Total CPU(secs)	408	320	501	625
	Mean CPU(secs)	14.6	9.1	8.8	13.3
	Mean elapsed time	220	267	1471	530
	CPU Utilisation	3%	2%	5%	5%

* System hangup occurred in 17/6 session which lasted about four minutes.

Table 7.2: Characteristics of the 1974 Terminal Workload

	20/5/74	12/6/74	17/6/74*	18/6/74
Session Start	9.29.08	9.35.26	8.25.29	9.32.29
Telex Start	9.32.42	9.36.32	9.10.42	9.33.54
Session End	13.00.20	13.01.00	12.59.56	13.00.08
No.of Terminal Sessions	184	198	204	175
Average Terminal Load	14.8	20.8	21.8	23.4
Maximum Terminal Load	25	34	39	40
Interactive CPU time used (secs)	1640	1880	1378	2320
Average CPU time/ session (secs)	8.9	9.5	6.8	13.3
Times No PP available	3597	5840	5924	7905
Times/hour no PP "	1003	1670	1690	2260
Telex CPU utilisation	3.3%	3.5%	4.8%	3.2%
Total(Batch+terminal CPU used)(secs)	8048	6858	5623	7071
Terminal User CPU utilisation	13%	15%	11%	19%
Batch User CPU utilisation	51%	41%	34%	39%
Total User CPU utilisation	64%	56%	45%	58%

* A system hangup occurred during 17/6 session which lasted for about 4 minutes.

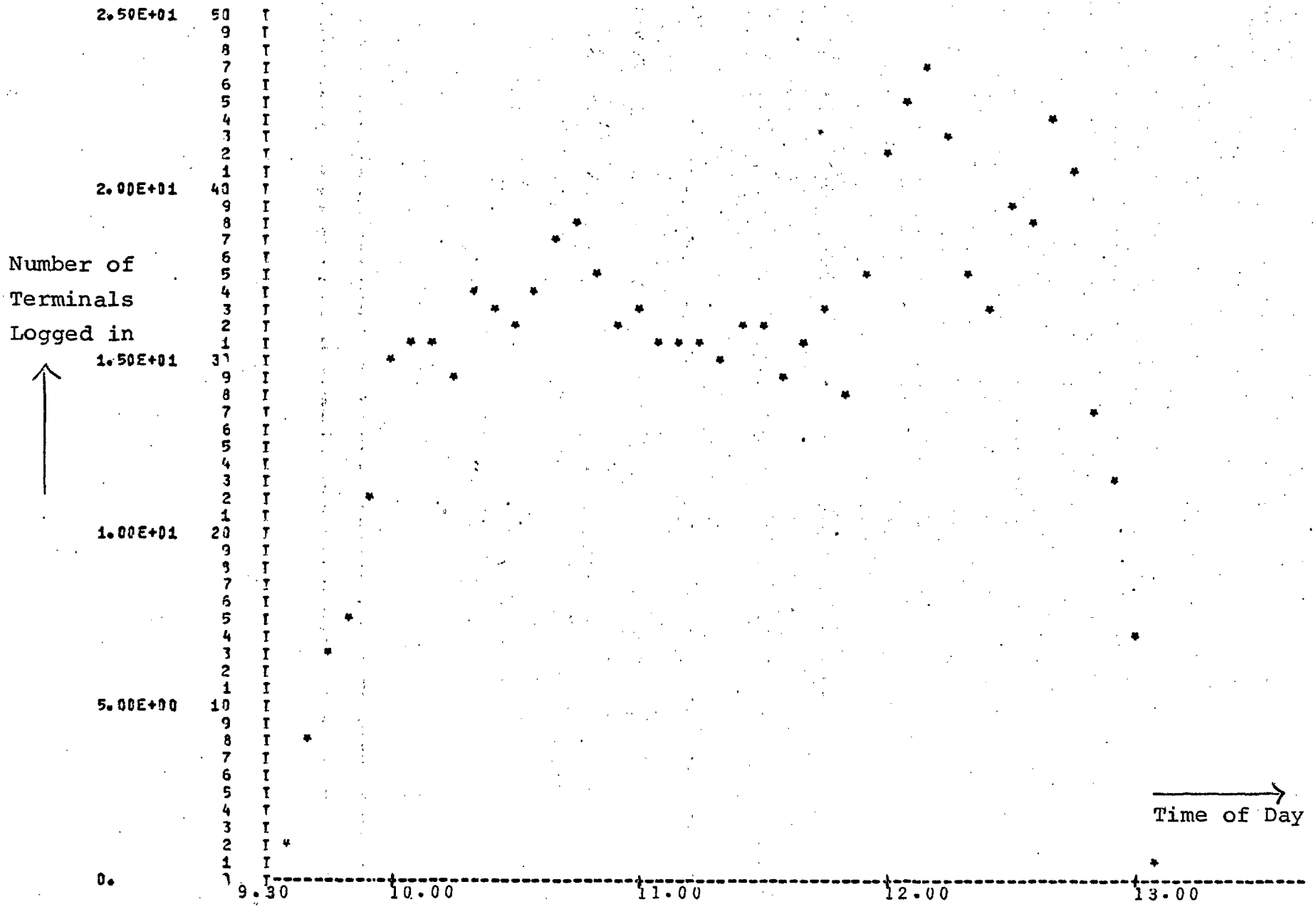


Figure 7.1: Average Terminal Load over 5 minute Intervals Plotted Against Time of Day (20/5/74am)

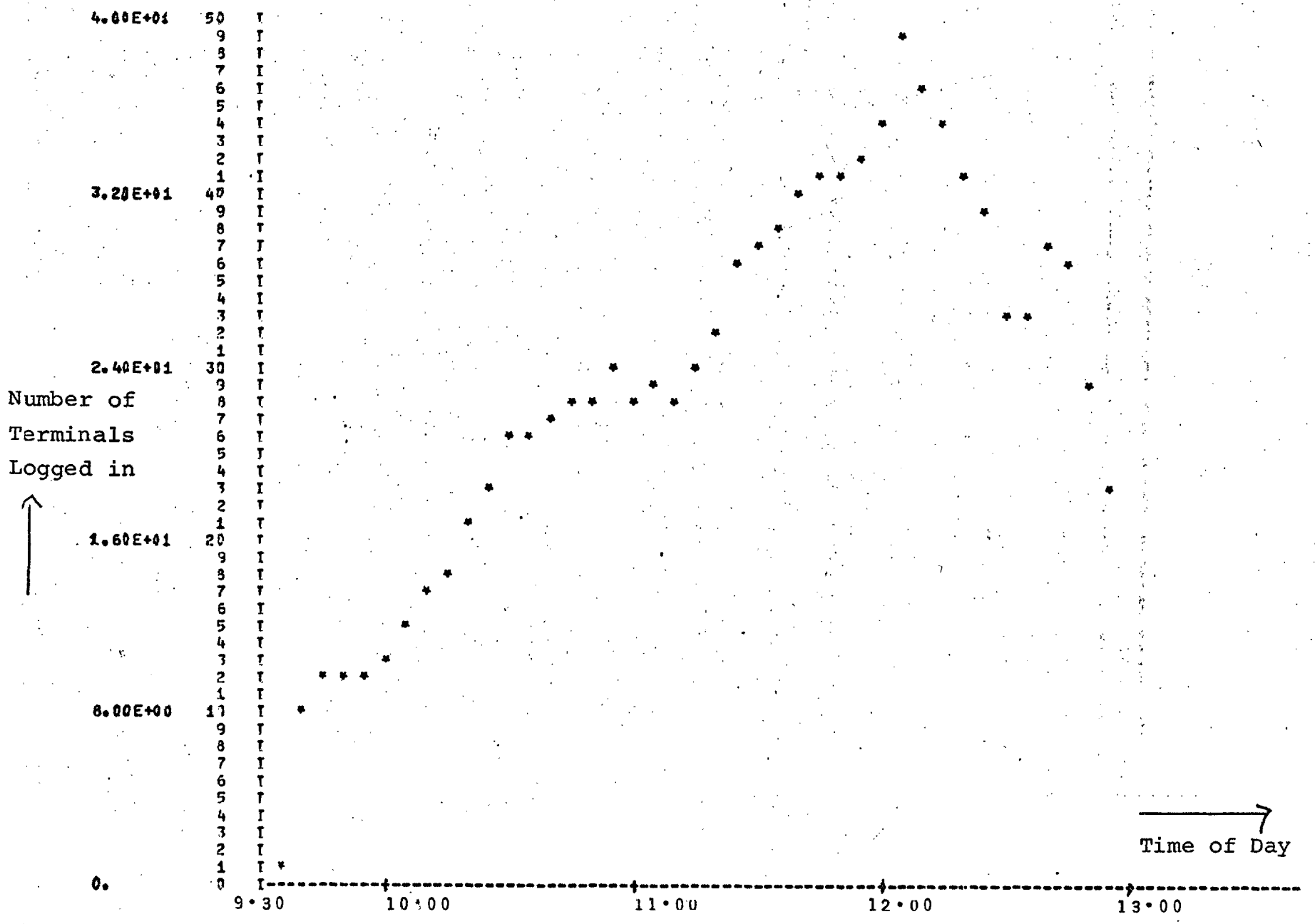


Figure 7.2: Average Terminal Load over 5 minute Intervals Plotted Against Time of Day (18/6/74 a.m.)

(PP) was available' when a PP was requested by Telex. This figure was lowest for 20/5 and highest for 18/6, when it was more than double that for 20/5.

PP activity is a measure of both system and I/O activity. PP activity can be high if the executing jobs are I/O bound, as it is the PPs which carry out all I/O operations. PP activity can also be high if system activity (such as rollin/rollout activity) is high, as most system functions are carried out by PPs. High terminal activity is liable to lead to high rollin/rollout activity and hence high PP activity. A previous performance analysis of the Kronos system (G6) showed that once the system becomes PP bound, performance degrades rapidly. It is interesting to note that the 20/5 session recorded the highest total CPU utilisation, while recording the lowest 'PP unavailable' figure. The 18/6 session recorded a much lower overall CPU utilisation while recording the highest 'PP unavailable' figure.

Table 7.3 shows the main characteristics of the two sessions that were modelled first, the mornings of 20/5 and 18/6. The description of the modelling is given in section 7.3. The analysis was first carried out on all the data, here called the untreated set. The analysis was then carried out again on a subset of this data, called the treated set. The reasons for excluding some observations from the data are given in section 7.3. The main difference in the two sets of data as shown in Table 7.3, is the reduction in mean elapsed time on 18/6 from 32.3 secs. to 21.8 secs.

A subsequent analysis was carried out on all four morning sessions, as described in section 7.5. Tables 7.4 and 7.5 show the main characteristics of the untreated and treated sets of data. Again the main difference in the two tables is the reduction in mean elapsed times.

Table 7.3: Characteristics of the 1974 Short Job Workload (Two Morning Samples)

		Untreated		Treated	
		20/5/74	18/6/74	20/5/74	18/6/74
Number of Jobs		401	467	336	373
Elapsed time (secs)	m.	23.7	32.3	19.2	21.8
	s.d.	21.8	42.9	14.2	15.0
CPU time (CPU) (secs)	m.	4.7	4.2	4.4	4.5
	s.d.	5.2	4.9	5.0	5.1
Disc records transferred (DPRU)	m.	286	219	269	219
	s.d.	378	156	350	154
No. Control Cards(NCC)	m.	3.9	5.3	3.9	4.7
	s.d.	2.5	4.9	2.5	4.0
No. jobs competing(AVB)	m.	4.2	7.1	4.2	6.7
	s.d.	2.0	2.6	2.0	1.9

Key: m. : mean
s.d. : standard deviation.

Table 7.4: Characteristics of the 1974 Short Job Workload
(Four Morning Samples Untreated)

		20/5/74	12/6/74	17/6/74	18/6/74
Number of Jobs		401	441	432	467
Elapsed time(secs)	m.	23.7	40.2	32.2	32.3
	s.d.	21.8	77.3	48.7	42.9
CPU time(CPU)(secs)	m.	4.7	4.0	4.5	4.2
	s.d.	5.2	4.8	5.3	4.9
Disc records transferred (DPRU)	m.	286	290	287	219
	s.d.	378	322	503	156
No. control cards(NCC)	m.	3.9	5.1	4.6	5.3
	s.d.	2.5	4.1	3.8	4.9
No. jobs competing(AVB)	m.	4.2	5.1	8.6	7.1
	s.d.	2.0	3.0	3.6	2.6
No. short jobs competing (AVJ)	m.	0.86	1.55	1.30	1.67
	s.d.	0.96	1.77	1.76	2.33

Key: m. : mean
s.d. : standard deviation

Table 7.5: Characteristics of the 1974 Short Job Workload
(Four Morning Samples Treated)

		20/5/74	12/6/74	17/6/74	18/6/74
Number of jobs		381	378	398	410
Elapsed time(secs)	m.	20.2	20.7	21.2	22.1
	s.d.	14.9	15.1	16.7	15.2
CPU time (secs)	m.	4.4	3.8	4.3	4.2
	s.d.	5.1	4.7	5.2	4.9
Disc records transferred	m.	269	265	230	215
	s.d.	335	256	182	151
No. control cards	m.	3.8	4.8	4.3	4.6
	s.d.	2.0	3.4	3.2	3.9
No. jobs competing	m.	4.2	4.6	8.4	6.6
	s.d.	2.0	2.8	3.6	2.1
No. short jobs competing with each job	m.	0.80	1.22	1.07	1.04
	s.d.	0.91	1.50	1.52	1.37
No. of jobs excluded		20	63	34	57

7.3 The First Models of the Short Job Workload

7.3.1 The Independent Variables

The dependent variable is the job elapsed time. The independent variables used in constructing the models, were obtained from both the Account and System Dayfiles. The independent variables fall into two groups:

(i) measures of a job's resource demands

(ii) measures of the load on the system, both batch and interactive, which a given job has to compete with.

The same measures of resource demand were used in constructing the models as for the 1973 models (Chapter 6), apart from one. The number of permanent file requests is no longer recorded in the Account Dayfile. In its place, the number of control cards (NCC) a batch job consists of, was derived from the System Dayfile. This is equivalent to the number of job steps in a job.

More extensive measures of the interactive load experienced by a given job were used in this analysis:

(a) Average number of terminals logged in during lifetime of this job (AVT):

$$AVT = \frac{\sum_{i=1}^{n} t_i}{t_e}$$

where t_e is a given job's elapsed time. n is the total number of terminal users who were logged in at any time while the job was in the execution phase. t_i is the time user i was logged in for, while the job was in the execution phase.

$$0 < t_i \leq t_e$$

(b) Rate of execution of terminal commands (RTC) while the job was in the execution phase:

$$\frac{\text{Number of terminal commands executed}}{t_e}$$

Measures of the batch load experienced by a given job were introduced for this analysis:

(a) Average number of batch jobs concurrently in execution with this job (AVB). This is not a measure of the level of multiprogramming, but rather a measure of the average number of jobs competing for Central Memory as well as the Central Processor. The multiprogramming level cannot be determined, as no indication of rollin/rollout is given in the Dayfile (section 5.5).

$$AVB = \frac{\sum_{j=1}^{m} t_j}{t_e}$$

m is the total number of batch jobs which were in the execution phase at any time while the given job was. t_j is the length of time job j was in the execution phase.

$$0 < t_j \leq t_e$$

(b) Rate of batch control and execution (RBC) while the given job was in the execution phase:

$$\frac{\text{Number of batch control cards executed}}{t_e}$$

7.3.2 The Initial Attempts

Two morning sessions, the 20/5 and 18/6, were used for constructing the first models of the short job workload which are shown in table 7.6. The models were considered unsatisfactory because the amount of variation explained by the model (R^2) is low, and the standard error of the residuals (s) is high. These values were particularly poor for the model of the 18/6 session, which also had a large intercept.

Table 7.6 also shows that no measures of terminal activity were selected by the Forward Selection Regression procedure. This suggests that no adequate measures of interactive load were available for selection. A measure of batch loading, the average number of batch jobs in execution with a job (AVB), was however selected for inclusion in both models. As might be expected, the more batch jobs competing for resources, the longer a job's elapsed time is likely to be.

7.3.3 Analysis of Residuals

The residual plots for the models of 20/5 and 18/6 are shown in figures 7.3 and 7.4 respectively. The general shape of these plots is similar to the residual plots obtained with the models of the 1973 workload, although it is more accentuated in the 18/6 plot than in the 20/5 plot. The main features are:

(i) There are a large number of small negative residuals and fewer larger positive residuals.

(ii) The large positive residuals occur for jobs with large elapsed times. Thus the model gives a poorer prediction for jobs with large elapsed times. The plots show that whereas six observations had residuals larger than 100 on 18/6, none did on 20/5.

Table 7.6: Initial Regression Models of Short Job Workload

Independent variable		20/5/74 a.m.	18/6/74 a.m.
CPU	r.c.	2.54	2.53
NCC	r.c.	1.18	4.14
AVB	r.c.	2.29	2.30
DPRU	r.c.	0.01	*
RBC	r.c.	*	0.36
Intercept		-5.11	-21.7
R^2		0.57	0.42
s		14.3	33.0
F		132	82
No. of jobs		401	467

Key:

*: not selected for inclusion in model

R^2: proportion of variation explained by model

s: standard error of residuals

F: F- statistic for significance of regression equation

r.c: regression coefficient

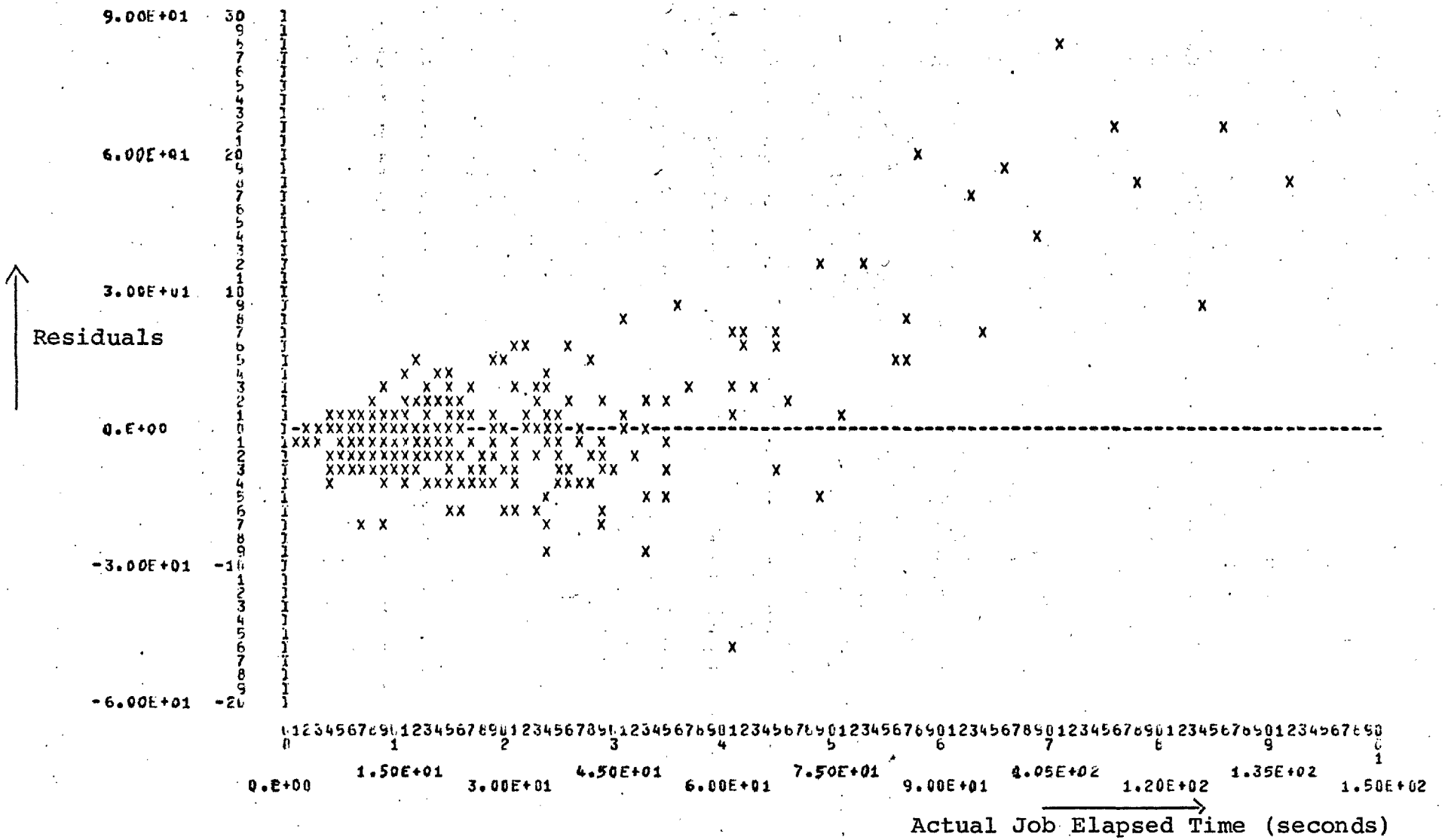


Figure 7.3: Residual Plot - Initial Model of Short Job Workload (20/5/74 a.m. session)

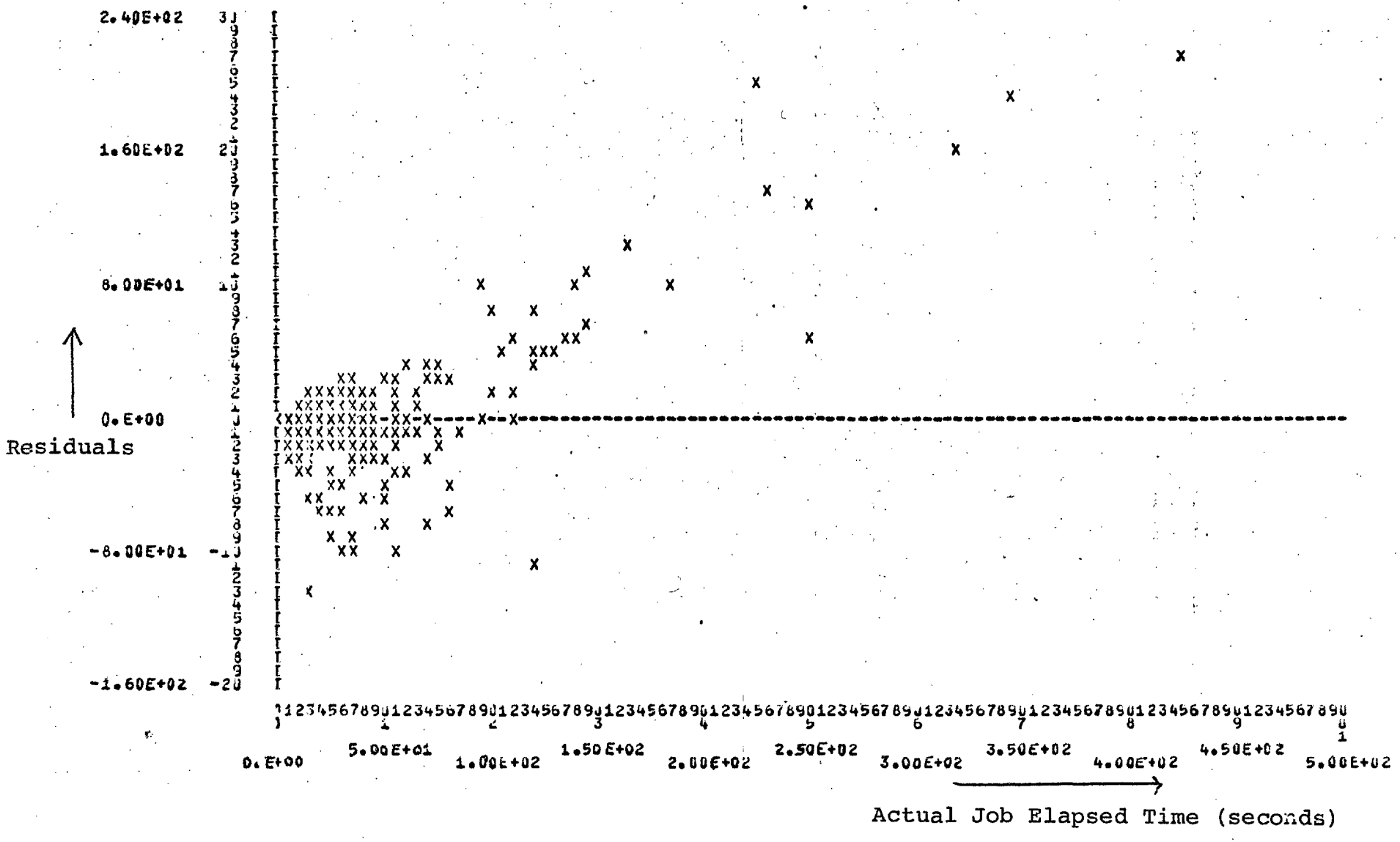


Figure 7.4: Residual Plot-Initial Model of Short Job Workload (18/6/74 a.m. session)

(iii) Furthermore, by examining the list of residuals, which is ordered by job termination time, it can be seen that the large positive residuals occur in groups. This indicates that there are certain periods in each session when the average elapsed time increases considerably. This occurred more often and for longer periods in the 18/6 session than in the 20/5 session.

It was argued in chapter 6 that a possible explanation of these features was the periodic fluctuations in the size of the interactive load. When the interactive load is high, batch jobs would be rolled out of CM (because of their lower priority) and would only be rolled in again when the interactive load subsided. Comparing the interactive loads for the two sessions (table 7.2), there are indeed indications that both the interactive load and overall system activity were considerably higher on 18/6 than on 20/5, a Monday morning session. Firstly, both the maximum and average number of terminals logged in were much higher on 18/6. Secondly, the interactive CPU utilisation was almost 50% higher on 18/6. Thirdly, the measure of overall system activity given by 'the number of times all PPs were busy' was more than twice as high on 18/6 than on 20/5.

7.3.4 Which way to go?

The analysis has shown that no adequate measures of interactive load were available for inclusion in the model. At this stage there seemed to be only two possible alternatives. The first was to decide that as apparently insufficient data was available to construct a satisfactory model, the attempt should be abandoned until more data was available. The new source of data would probably have to be a software monitor which would require time and manpower to implement.

The second alternative also admitted that a satisfactory model, which explained all situations from light to heavy system loads, was impossible to construct from the existing data. However, this alternative suggested that it might still be possible to construct a satisfactory model for light and moderate loads on the system. The most encouraging indication that this might be possible, was the difference in the models for 20/5 and 18/6. In the 20/5 model, 58% of the variation in job elapsed time (R^2) was explained by the model, whereas only 40% was explained by the 18/6 model. The larger R^2 in the 20/5 model is due to the smaller residuals for that model. The standard error of residuals for 20/5 was 14.3, compared with 33.0 for 18/6 (table 7.6).

The decision was therefore taken to exclude the groups of observations where large positive residuals had occurred and to construct models using the remaining observations. This was felt to be a justifiable move, because as shown by the analysis of residuals, the large positive residuals occurred in groups at certain periods of the day. This indicated that during these periods the system was behaving differently. This will be confirmed later in section 7.4, where it is shown that during the periods in question the system was heavily loaded.

7.3.5 Models with Large Residuals Excluded

Models were built for the 20/5 and 18/6 sessions using data from which large residuals had been excluded. These models showed a marked improvement over the previous set. However, different independent variables were selected for inclusion in the two models. This is apparently a common feature of all the stepwise regression procedures, particularly when some of the independent variables are correlated with each other. This feature is one of the main disadvantages of these procedures (D1).

To enable a direct comparison of the models to be made, models containing the same independent variables were constructed for each session. This was achieved by forcing into each model the same set of independent variables, which were:

(a) CPU - the CPU time used by the job

(b) NCC - the number of control cards (job steps) in the job

(c) AVB - the average number of batch jobs concurrently in execution with this job.

Models for the 20/5 and 18/6 sessions, and a model for both sessions combined, were constructed using the method just described. The main features of the models are shown in table 7.7. A number of points may be drawn by comparing these models with the initial models (7.3.2 and table 7.6).

1. In both the 20/5 and 18/6 models R^2 has improved substantially; from 0.58 to 0.78 for the 20/5 model, and from 0.40 to 0.76 for the 18/6 model (comparing table 7.6 with table 7.7).

2. The standard error of the residuals (s), has been reduced considerably in both models. In the 20/5 model, s has been reduced from 14.3 to 6.6 and in 18/6 model, s has been reduced even more sharply from 33.0 to 7.4.

3. The intercepts have been reduced considerably especially for the 18/6 model, and they are small compared with the mean of the job elapsed time (table 7.3).

4. Comparing the regression coefficents of the independent variables for the 20/5 and 18/6 models, it can be seen that (table 7.7):

Table 7.7: Regression Models of Short Job Workload
(Large residuals excluded)

Independent Variable		20/5/74	18/6/74	Combined
CPU	r.c.	2.32	2.30	2.30
	s.e.	0.07	0.08	0.05
	t	33.2	28.8	46.0
NCC	r.c.	1.14	1.29	1.22
	s.e.	0.15	0.10	0.08
	t	7.6	12.9	15.3
AVB	r.c.	1.64	0.94	1.06
	s.e.	0.18	0.20	0.11
	t	9.1	4.7	9.7
Intercept		-2.17	-0.84	-0.71
R^2		0.78	0.76	0.77
s		6.6	7.4	7.1
F		399	390	772
No. of jobs		336	373	709

Key:
- r.c. : regression coefficient
- s.e. : standard error of regression coefficient
- t : t-statistic for significance of regression coefficient
- R^2 : proportion of variation explained by model
- s : standard error of residuals
- F : F-statistic for significance of regression equation

(a) the coefficients of the CPU variable (2.32 and 2.30) are almost identical.

(b) the coefficients of NCC (1.14 and 1.29) are reasonably close.

(c) the coefficients of AVB (1.64 and 0.94) show a greater variation.

It is clear that there is a much closer agreement between the regression coefficients than has been the case previously.

5. The residual plots for the 20/5 and 18/6 models are shown in figures 7.5 and 7.6 respectively. Comparing these with figures 7.3 and 7.4, it can be seen that although there is still a tendency for the large residuals to occur with large job elapsed times, this effect is not as marked as before, particularly for the 18/6 session. In both figures 7.5 and 7.6 there appear to be two bands sloping upwards from left to right. This is probably due to a characteristic of the short job workload. Most jobs have a small CPU demand and fall in the leftmost band. However, a sizeable minority of jobs run till they time trap (just over 16 CPU seconds). The contribution due to CPU time, as predicted by the model, will be virtually identical for all these jobs. These jobs fall into the second band. The variation in predicted elapsed time for these jobs is due to the contribution of the other two variables (NCC and AVB).

7.4 The Interaction between the Workload and the System

7.4.1 The Jobs Excluded

Constructing the much improved models described in the last section, showed that there was no inherent reason

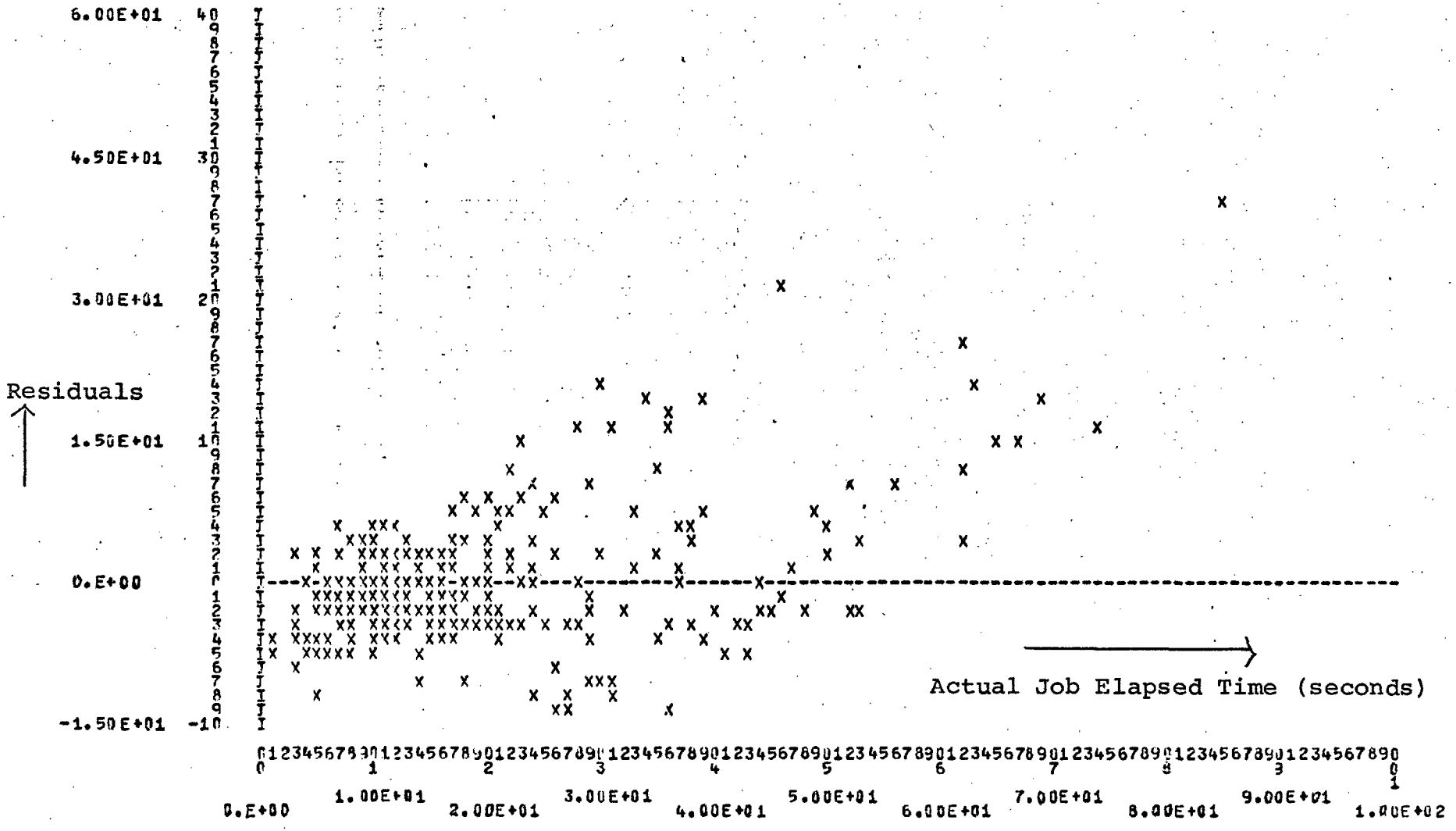


Figure 7.5: Residual Plot - Model of Short Job Workload - Large Residuals Excluded

(20/5/74 a.m. session)

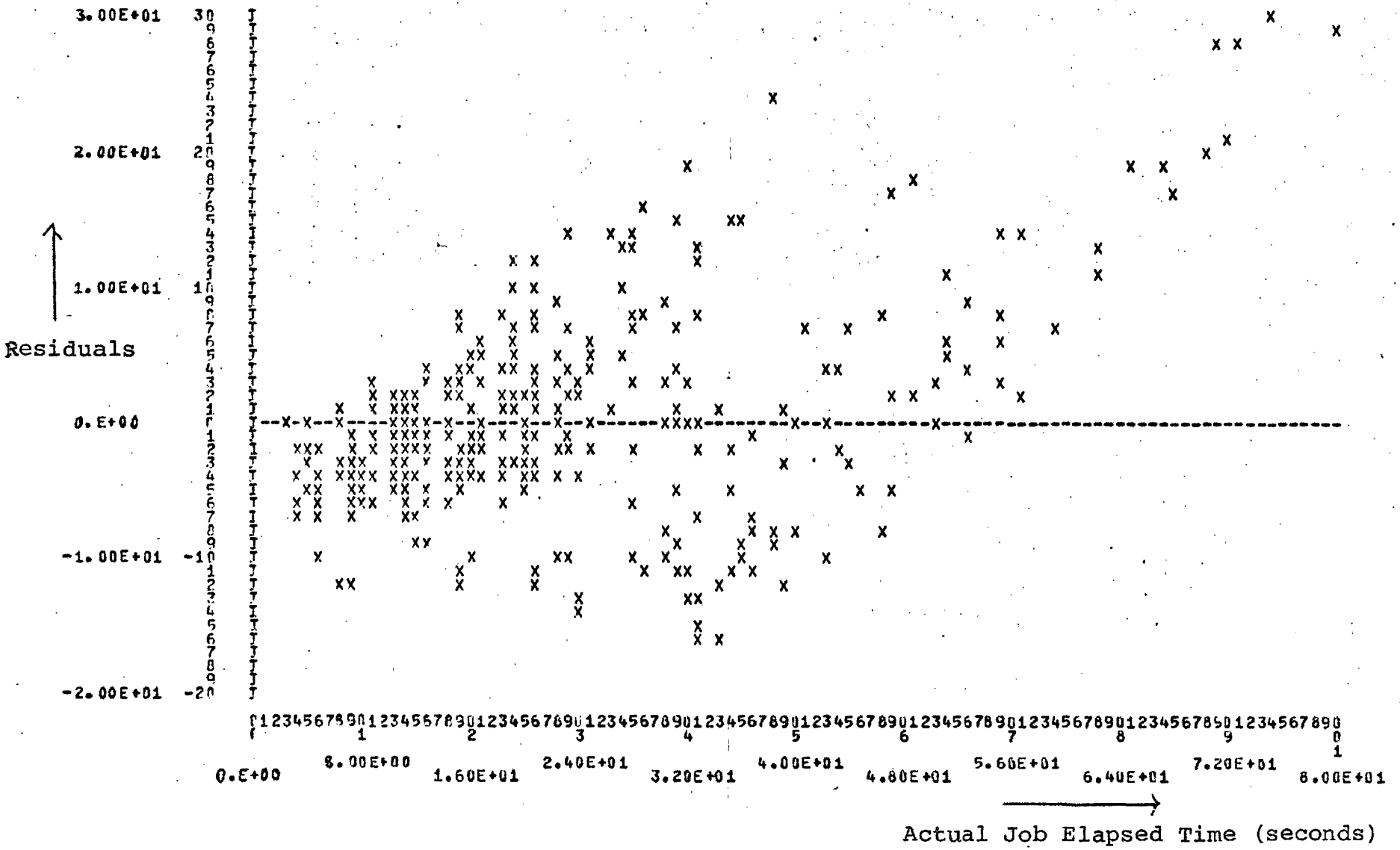


Figure 7.6: Residual Plot: Model of the Short Job Workload - Large Residuals Excluded (18/6/74 a.m. session)

why satisfactory regression models of the IC system at the workload level, could not be built. The question still remained however: why did a number of jobs have to be excluded from the sample to achieve this? Was it entirely due to differences in the interactive workload or could there also have been some other influence so far undetected?

The excluded jobs fell into three main categories:

1. A small number of jobs appeared to hang at a control point until they were dropped by the operator. This was revealed in the System Dayfile by an 'Operator Drop' message for these jobs at job termination.

2. A few jobs appeared to be uncharacteristic in their resource demands. In some cases, the number of control cards (NCC) was very high. In others, the number of disc records transferred (DPRU) was very high. Such observations should be excluded; otherwise, as shown in chapter 6, they may distort the model considerably. Other jobs did not appear to display any unusual characteristics, as judged by the data recorded in the Dayfile. Nevertheless, in more than one case, the same job (identifiable by job number, control cards used and resource utilisation) was run at different times of the morning, each time taking much longer than the model predicted. This definitely suggests that the job was uncharacteristic. For example it might have been an I/O bound job making a large number of I/O requests (not recorded in the Dayfile) but only transferring a few records each time, and so resulting in a comparatively small number of disc records transferred.

3. Most of the jobs with large residuals fell into this third category. This was particularly noticeable in the 18/6 sample. At two different times of the

morning and particularly on the second occasion around 12.15 p.m., the average job elapsed time increased substantially.

The two periods in question had one feature in common; the number of batch jobs in execution concurrently (AVB) was high. However there was a third occasion in this session when AVB was high, even though the elapsed time was normal.

7.4.2 The Reason for the Large Job Elapsed Times

The reason for this apparent discrepancy was found by a close scrutiny of the 18/6 System Dayfile for the three periods when the average batch load was high. On the two occasions when the average short job elapsed time increased substantially, the number of short jobs concurrently in execution was high. On the third occasion, when the elapsed time did not increase noticeably (although the number of jobs concurrently in execution was high) the number of short jobs concurrently in execution was low.

This immediately suggested that the memory scheduling algorithm had changed since the 1973 analysis, when all batch jobs executing concurrently were treated in the same manner. It now seemed that short jobs had priority over other batch jobs in the competition for Central Memory (CM). Thus on two occasions when the short job load was high, the average short job elapsed time increased substantially. It was indeed confirmed later by the I.C. Computer Centre management that short jobs had priority over other jobs in the competition for CM.

The reason for the large increase in elapsed time is that CM is a scarce resource on the IC system. On average only 2 or 3 batch jobs may be co-resident in CM. Therefore if the short job load is high (3 upwards), some

short jobs are likely to be rolled out at some stage in their execution. The bigger the load on the system, the longer, on average, jobs will be rolled out. The longer a job is rolled out, the longer its elapsed time will be.

In the case of the 18/6 sample, the bottleneck at around 12.15 p.m. was accentuated by the fact that it was caused by the influx of a substantial number of batch jobs (all with the same job number and submitted from terminals) with comparatively similar characteristics: small CPU time, large number of control cards, and apparently testing a mathematical program library. These jobs appeared to be I/O bound, and when run concurrently caused the bottleneck just described. When a similar set of jobs were run in the afternoon, the elapsed time was even worse, increasing from around 400 seconds to over 3000 seconds, and in two cases over 4000 (i.e. more than an hour to run a one CPU second job).

As pointed out in sections 7.2 and 7.3, both the terminal load and overall system activity were higher on 18/6 than on 20/5. Although the terminal load, as measured by the number of terminals logged in, may vary comparatively slowly (figures 7.1 and 7.2), there are likely to be bursts of terminal user activity during periods when a substantial number of terminal users submit programs to be compiled or run interactively. These terminal job runs at control points making comparatively high resource demands (section 4.5.2). Since they have a higher priority than short batch jobs, a buildup of rolled out short jobs will occur (D3), leading to longer elapsed times.

Thus the two main causes of the longer elapsed times experienced by short jobs are:

(a) Bursts of terminal user activity resulting in short jobs being rolled out.

(b) A substantial number of short jobs in competition with each other resulting in some short jobs being rolled out.

(b) may be caused or accentuated by (a). However (b) can occur of its own accord if a substantial number of short jobs are submitted in close succession, and in particular if they make comparatively large resource demands.

7.4.3 The Scheduling Algorithms

Apart from going a long way towards explaining in what manner the models described in 7.3.2 are deficient, the analysis also reveals apparent weaknesses in the system job and memory scheduling algorithms. Firstly, the job scheduling algorithm sets no limit on the number of short jobs that may be scheduled for execution concurrently. As a result, considerable delays in elapsed time can be experienced by the competing jobs, when the load is high.

The second apparent weakness is in the memory scheduling algorithm and is because:

(a) All short jobs have the same access priority to CM.

(b) When 1SJ (the PP program in charge of both job and memory scheduling) searches the File Name Table (FNT) for jobs in Rollout or Input states, it always starts at the top of the table. As a result, rolled out jobs near the top of the table have a better chance of being scheduled for rollin than jobs with the same priority, but further down the table. Examining the System Dayfile revealed that this can have a considerable effect on a job's elapsed time, once a job has been rolled out. Jobs can be identified as being rolled out when no message (e.g. initiation of control card execution) is recorded in the System Dayfile for a substantial period of time. When a number of jobs are rolled out, it is

noticeable that there is no particular order in the re-activation of jobs and that for some unfortunate jobs, many minutes can pass before they are reactivated.

The IC Computer Centre have since amended the algorithm used by 1SJ in searching the FNT. 1SJ now starts a search of the FNT at the place it previously stopped at.

7.5 Regression Models of the Morning Workload

7.5.1 Introduction

The analysis and modelling described in sections 7.3 and 7.4 provided a deeper insight into the interaction between the workload and system. To consolidate further, a further analysis was carried out. This included two further morning sessions, the 12th and 17th June, in addition to the 20th May and 18th June used previously.

In the light of experience, some additional variables were made available for selection, as follows:

(a) It was now known that the important influence on a short batch job's elapsed time is not the total number of batch jobs concurrently in execution (AVB), but rather the number of short batch jobs concurrently in execution. An additional variable was therefore used which represents the average number of short jobs in execution with a given job (AVJ).

(b) Use of a load term in the model such as the average number of short jobs in execution (AVJ) implies that a given load has an equal effect on job elapsed time, no matter what the resource demands of the job are. It seems more likely that a job with large

resource demands would suffer a bigger delay than a job with small resource demands. Since CPU time is the variable which made the most significant contribution to the models described in section 7.3, a term of the form CPU * AVJ was made available for selection.

(c) The variable representing the average number of terminals logged in during a job's lifetime (AVT) made no significant contribution to the models described in section 7.3. However, following the same argument as in (b), it was felt that a variable of the form CPU * AVT might make a more significant contribution.

7.5.2 The Initial Models

In spite of introducing independent variables relating to AVJ, the initial models were still not satisfactory. However they did show an improved fit over the initial models discussed in section 7.3. The major problem is that:-

(a) simple functions (such as the average number of jobs in execution) are being used to model a complex function (the delay in job elapsed time when the load gets high, resulting in jobs being rolled out).

(b) these simple functions are not a sufficiently accurate representation of the complex function.

A job may be rolled out for one of four reasons:

(i) A higher CM priority job is scheduled for execution (e.g. a timesharing user).

(ii) The job requests additional memory (CM) which is unavailable.

(iii) The job exceeds its CPU time slice. If a batch job has used more than this figure (currently set to 4 CPU seconds) it is liable to be rolled out.

(iv) The job exceeds its CM time slice. If a batch job has been resident in CM for more than this figure (currently set to 200 seconds), it is liable to be rolled out.

The profile of the short job workload is such that a short job is much more liable to be rolled out for reason (iii) than (iv).

Each time a job is rolled out, its elapsed time is likely to increase in a stepwise fashion. As pointed out in section 7.4, once a job has been rolled out, it can be subjected to severe delays, if it is low down in the FNT table. Furthermore, the performance analysis of the Kronos system described in (G6) has shown that when the system gets overloaded, non-linearities are introduced into the system and jobs may experience long delays. To get good measures of the system in an overloaded state requires considerably more performance data than is currently available from the Dayfile (section 5.5).

7.5.3 Constructing the Models

For the same reasons given in section 7.3, it was decided that models should be constructed from which the largest outliers had been excluded. However a different approach was adopted for eliminating outliers. An iterative approach was taken with the successive elimination of a few outliers at a time, until a good fit was obtained. A number of runs were carried out for each model, in which large residuals were successively deleted. Gradually an improved R^2 and reduced standard error of residuals were obtained.

There were a number of interesting points in the construction of these models:

(a) In all four models, the first variable introduced always included a CPU factor. Usually, the first variable was either CPU or CPU * AVT. These two variables are highly correlated: in all four sessions the correlation coefficient was 0.9 or more. Because of this, the inclusion of one variable in the model tended to exclude the other.

(b) The number of observations that had to be excluded in order to obtain a good fit varied considerably from one sample to the next. Thus in 20/5, only 22 observations were eliminated, while in 12/6, 63 observations were eliminated.

(c) It has already been described in section 7.3, how one period of the 18/6 session experienced severe degradation. To get a good overall fit, it was necessary to exclude all jobs that were executed in that period.

(d) As pointed out in section 7.2, a hangup occurred for a period of about four minutes during the 17/6 session. Jobs held up by this blockage were excluded in the construction of the 17/6 model.

(e) The short job load was highest on the 12/6 session. 63 observations had to be deleted (out of 441) to obtain a good fit. Most of the deleted observations corresponded to jobs which fell into two periods when the short job load was high. There were also three jobs which hung at control points and were dropped by the operator.

(f) A satisfactory regression model was constructed more easily for the 20/5 session. As pointed out in section 7.2 the 20/5 session was the most CPU bound and least PP bound session. It also had the least terminal activity and the least concurrent batch activity. Higher batch and terminal activity lead to greater rollin/rollout activity which in turn means greater PP activity and consequent job delays. This explains why the 20/5 model was the least difficult to construct.

7.5.4 The Regression Models

7.5.4.1 Introduction

Models were eventually built for each of the four sessions. In all four models, R^2 was 0.7 or better. As before, however, different independent variables were selected for inclusion in the models. To enable a direct comparison of the models to be made, the same set of independent variables were forced into each model. By this means, different versions of each model were constructed. For each version, a combined model was also constructed, which used data pooled from all four samples. Tables 7.8, 7.9 and 7.10 show three different versions of the four models as follows

Table 7.8 - Independent variables are CPU, NCC and AVJ
Table 7.9 - Independent variables are CPU, NCC and CPU * AVJ
Table 7.10 - Independent variables are CPU * AVT, NCC and CPU * AVJ

Building models with only three independent variables meant that in some cases R^2 was reduced by a few percent below 0.70.

Table 7.8: Regression Models of Short Job Workload
First Version

Independent Variable		20/5/74	12/6/74	17/6/74	18/6/74	Combined
CPU	r.c.	2.31	2.21	2.25	2.24	2.26
	s.e.	0.08	0.10	0.09	0.08	0.04
	t	28.9	22.1	25.0	28.0	56.5
NCC	r.c.	1.53	1.37	1.58	1.32	1.42
	s.e.	0.19	0.14	0.15	0.10	0.07
	t	8.1	9.8	10.5	13.2	20.3
AVJ	r.c.	3.91	1.86	3.15	3.05	2.81
	s.e.	0.43	0.30	0.31	0.29	0.16
	t	9.1	6.2	10.2	10.5	17.6
Intercept		1.09	3.44	1.28	3.33	2.44
R^2		0.75	0.67	0.68	0.70	0.72
s		7.5	8.7	9.5	8.1	8.5
F		373	258	282	346	1217
No. of jobs		381	378	398	410	1567

For key to abbreviations refer to table 7.7.

Table 7.9: Regression Models of Short Job Workload
Second Version

Independent Variable		20/5/74	12/6/74	17/6/74	18/6/74	Combined
CPU	r.c.	1.73	1.65	1.73	1.79	1.75
	s.e	0.09	0.11	0.10	0.10	0.05
	t	19.3	15.0	17.3	17.9	35.0
NCC	r.c.	1.48	1.47	1.52	1.31	1.42
	s.e.	0.19	0.13	0.15	0.11	0.07
	t	7.8	11.3	10.1	11.9	20.3
CPU*AVJ	r.c.	0.71	0.53	0.49	0.41	0.50
	s.e.	0.07	0.06	0.05	0.06	0.03
	t	10.2	8.8	9.8	6.9	16.7
Intercept		4.56	5.44	4.88	6.78	5.51
R^2		0.76	0.70	0.69	0.68	0.70
s		7.3	8.3	9.3	8.6	8.5
F		400	291	291	292	1237
No. of jobs		381	378	398	410	1567

For key to abbreviations refer to table 7.7

Table 7.10: Regression Models of Short Job Workload
Third Version

Independent Variable		20/5/74	12/6/74	17/6/74	18/6/74	Combined
CPU*AVT	r.c.	0.10	0.08	0.07	0.06	0.07
	s.e.	0.005	0.005	0.004	0.004	0.002
	t	20.0	16.0	17.5	15.0	35.0
NCC	r.c.	1.53	1.60	1.67	1.28	1.50
	s.e.	0.19	0.13	0.14	0.11	0.07
	t	8.1	12.3	11.9	11.6	21.4
CPU*AVJ	r.c.	0.71	0.40	0.35	0.46	0.45
	s.e.	0.07	0.06	0.05	0.06	0.03
	t	10.1	6.7	7.0	7.7	15.0
Intercept		4.80	5.21	4.55	7.63	6.07
R^2		0.76	0.71	0.72	0.67	0.69
s		7.3	8.1	8.9	8.7	8.6
F		399	311	332	279	1168
No. of jobs		381	378	398	410	1567

For key to abbreviations refer to table 7.7

7.5.4.2 Relevance of the Independent Variables

Before discussing the models in detail, this section considers the relevance of each of the variables selected in the final versions of the model:

(i) Central Processor (CPU) Time

There is an obvious relationship between a job's elapsed time and the CPU time it uses. The more CPU time a job uses, the longer its elapsed time is likely to be. The relationship between elapsed time and CPU time is shown in figure 7.7 for the 18/6 session.

(ii) Number of Control Cards (NCC)

This is equivalent to the number of job steps in each job. Before a job step may be executed a PP program, the Job Advancer (1AJ), interprets the control card and initiates the job step by loading the appropriate PP or CM program. Delays are liable to occur for the following reasons:

(a) If the system is heavily loaded, there may be a delay before 1AJ is loaded into a PP.

(b) 1AJ has to interpret the control card and initiate job step execution. The program that is to be loaded (e.g. compiler or relocatable binary loader) will probably reside on disc. Further delays may be experienced before the program is loaded off disc.

(c) A new job step will often require a change in memory allocation. If less memory is required,

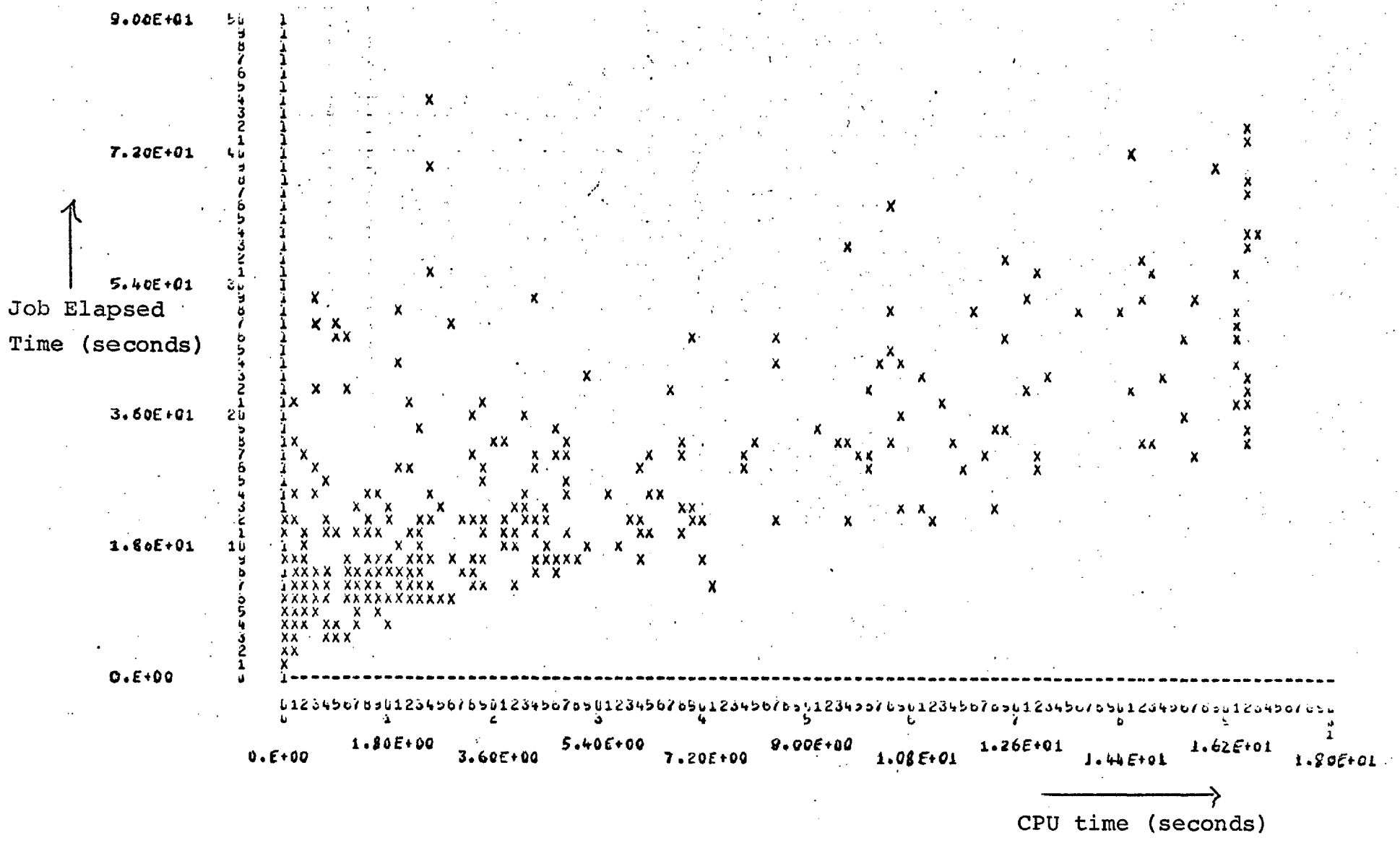


Figure 7.7: Short Job Workload - Variation of Job Elapsed Time with CPU Time

the request will be rapidly satisfied. If an increase in memory is required and sufficient memory is available, the request will also be rapidly satisfied. However if insufficient memory is available, the job is likely to be scheduled for rollout, particularly if there are other jobs with the same or higher priority competing for memory.

Thus a job with a larger number of job steps is more likely to experience delays than a job with identical resource demands but fewer job steps. The delay due to this is likely to be especially important for short jobs whose resource demands are comparatively small. The coefficient of the NCC variable can thus be considered as a measure of the system overhead to initiate a job step.

(iii) Load Variables

As pointed out earlier, the elapsed time a short job experiences can be seriously extended if the terminal activity is high or if there are a large number of competing jobs. This is the reason for the importance of the load variables. Three load variables were used in the models. One is the average number of short jobs each job has to compete with (AVJ). An alternative is CPU * AVJ, in which the short job load is weighted by the job's CPU time requirements. The third, CPU * AVT is a measure of terminal load, weighted by a job's CPU time requirements. Further comments are made on the use of these variables in the next three subsections.

7.5.4.3 The First Version (CPU, NCC, AVJ) of the Models

In this version of the model, the three independent variables are not correlated with each other. Thus, the regression coefficient of each variable reflects the only contribution made by that variable to the model. To illustrate this, consider the CPU and AVJ coefficients in the 20/5 model. The model is constructed using the Forward Selection Procedure (Chapter 6), in which the three independent variables are forced in one at a time.

1st step	$y = 9.58 + 2.40$ CPU	$R^2 = 0.65$
2nd step	$y = 6.42 + 2.42$ CPU $+ 3.85$ AVJ	$R^2 = 0.71$
3rd step	$y = 1.09 + 2.31$ CPU $+ 3.91$ AVJ $+ 1.53$ NCC	$R^2 = 0.75$

It can be seen from the above example, that the regression coefficients of the CPU and AVJ variables do not fluctuate much with the introduction of each new variable.

The models of the four sessions, together with a model of the combined sessions, are shown in table 7.8. The main features of the models are:

(a) In the four models, R^2 varies between 0.67 and 0.75. These values of R^2 are considered satisfactory, as they mean that between two thirds and three quarters of the variation in job elapsed time is explained by the models.

(b) There is a close agreement between the regression coefficients of the CPU variable.

(c) There is also a reasonably close match between the regression coefficients of the NCC variable.

(d) There is a much greater variability in the coefficient of the load variable, AVJ.

A limitation in the use of a load variable such as AVJ is that no account is taken of the job's resource demands. In the next version of the models, an attempt was made to rectify this by using a CPU * AVJ term.

7.5.4.4 The Second Version (CPU, NCC, CPU*AVJ) of the Models

In this version, the load variable is represented by CPU * AVJ (table 7.9). The three independent variables are now no longer uncorrelated, as CPU * AVJ is correlated with CPU. In the four sessions, the correlation coefficient between CPU and CPU * AVJ was 0.5 or more. Comparing tables 7.8 and 7.9 it can be seen that, replacing the AVJ term by CPU * AVJ, has a comparatively small effect on the NCC coefficients, but a much larger effect on the CPU coefficients. For example, in the 20/5 model, the CPU coefficient is reduced from 2.31 in table 7.8 to 1.73 in table 7.9, whereas the NCC is only reduced from 1.53 to 1.48.

The main features of the models are:

(a) Comparing the first and second versions of each model (tables 7.8 and 7.9), it can be seen that there is little difference in R^2 and s. However the intercepts are somewhat bigger in the second version of the models.

(b) Comparing the regression coefficients of the four models (table 7.9), it can be seen that there is a relatively close match between the coefficients of the CPU and NCC variables.

(c) As before, however, there is a greater variability in the regression coefficient of the load variable, CPU * AVJ.

The load term can still not be considered satisfactory. One obvious omission is the terminal load, which is present and varying throughout each session. As observed earlier, there appears to be no suitable variable representing terminal load available.

7.5.4.5 The Third Version (CPU * AVT, NCC, CPU * AVJ) of the Models

An attempt was made to introduce a variable which is related to the terminal load by using CPU * AVT. However this variable is very highly correlated with CPU. The correlation coefficient was 0.9 or more for the four sessions. It is thus apparent that the CPU factor is by far the major contributor to the CPU * AVT term in the model. Indeed, using the Forward Selection Procedure, the introduction of CPU * AVT into the model had the effect of excluding CPU (and v.v.).

Table 7.10 shows the third version of the models for each session. Like the previous two, this version has a satisfactory R^2, but like the second version it has a larger intercept. This version is less satisfactory than the previous two, however, because there is a greater variability in all three sets of regression coefficients.

7.5.4.6 Analysis of Residuals

The analysis of residuals for these models did not reveal any major differences from the analysis carried out on the previous models of the short job workload des-

cribed in section 7.3. The residual plots against actual elapsed time for each version of the four models are similar in shape to those shown previously in figures 7.5 and 7.6. As an example, the residual plot for the 12/6 model (first version) is shown in figure 7.8.

7.5.5 Comparison of the Models

In all three versions of the models, R^2 is greater than 0.67, which is considered satisfactory. In the first and second versions, the regression coefficients agree more closely than in the third. In all three versions, there is a greater fluctuation in the regression coefficients of the load variables.

Of the first and second versions, the points favouring the first version (with AVJ) are:

(a) the intercepts are smaller

(b) the three independent variables are independent of each other. Hence the regression coefficients are independent measures of the contribution made by each of the variables to the model.

The main point favouring the second version (with CPU * AVJ) is that the load variable includes a measure of resource demand. However the fluctuation of the regression coefficients indicates that the CPU * AVJ variable can still not be considered an adequate measure of the load on the system. The model also suffers from the disadvantage that the contribution by CPU is now split between two terms, CPU and CPU * AVJ. The first two versions both suffer from the disadvantage of not having a measure of terminal loading.

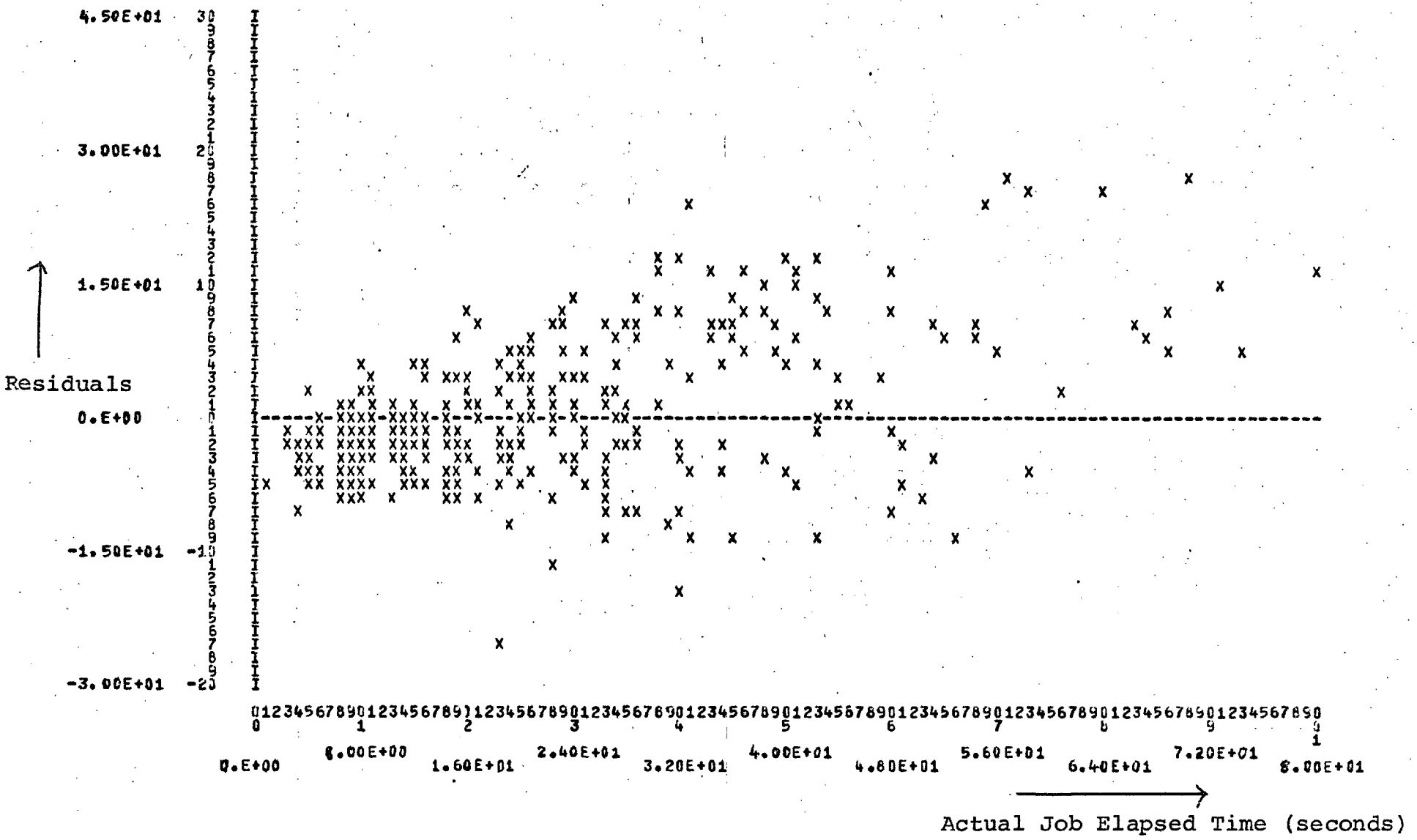


Figure 7.8: Residual Plot - Model of Short Job Workload (12/6/74 a.m. session)

The higher R^2 in the 20/5 model for all three versions of the models is almost certainly due to the lower terminal load during that session.

7.6 Regression Modelling of the Afternoon Workload

7.6.1 Analysis of the Afternoon Sessions

Two afternoon sessions were also analysed, the 17/6 and 18/6 sessions. The main characteristics of the batch and interactive workload for these sessions are shown in tables 7.11 and 7.12 respectively.

Table 7.11 shows that many characteristics of the batch workload were remarkably similar on the two afternoons. The total number of jobs executed was close, although the breakdown into short and long jobs was different. The batch CPU utilisation, as well as the short and long job CPU utilisation were all virtually identical for the two sessions. However, the mean elapsed time for short jobs on 18/6 was more than double that for 17/6.

A close examination of the System Dayfile for the 18/6 session revealed that at one period of the afternoon, a large number of short jobs with similar characteristics were submitted and scheduled for execution over a short interval of time. This resulted in a system bottleneck. A similar event occurred in the morning session of the same day and was reported in section 7.4.2. However, the afternoon bottleneck appears to have been considerably more severe. In the afternoon, the elapsed time of some of these jobs was over 3000 seconds, and in one case over 4000 seconds. During this period, the number of batch jobs executing concurrently peaked at 33. The mean elapsed time for short jobs was 104 seconds in this session, compared with 32 seconds for the 18/6 morning session and 24 seconds for the 20/5 morning session.

Table 7.11: Characteristics of the Afternoon Batch Workload

Class	Characteristic	17/6/74	18/6/74
All Jobs	No. of jobs	976	995
	Total CPU time (secs)	7820	7870
	Mean CPU time/job (secs)	8.0	7.9
	Batch CPU utilisation	36.2%	36.4%
Short Jobs	No. of jobs	872	831
	Total CPU time (secs)	4360	4410
	Mean CPU time/job (secs)	5.0	5.3
	Mean elapsed time (secs)	45.7	104.4
	CPU utilisation	20.2%	20.4%
Long Jobs	No. of jobs	104	164
	Total CPU time (secs)	3460	3460
	Mean CPU time/job (secs)	33.7	31.1
	Mean elapsed time (secs)	1262	833
	CPU utilisation	16%	16%

7.12: Characteristics of Afternoon Terminal Workload

	17/6/74	18/6/74
Session start	14.04.04	14.06.17
Session end	20.00.00	20.00.03
No. of terminal sessions	322	294
Average Terminal Load	22.4	21.7
Maximum Terminal Load	40	36
Interactive CPU time used (secs)	4270	3320
Average CPU time/session (secs)	13.3	10.3
Times no PP available	17,071	15,846
Times/hour no PP available	3,570	2,640
Telex CPU utilisation	4.7%	3.5%
Terminal User CPU utilisation	19.8%	15.4%
Batch User CPU utilisation	36.2%	36.4%
Total User CPU utilisation	56.0%	51.8%

Table 7.12 shows the main characteristics of the terminal workload on the two afternoons. The terminal CPU utilisation was higher on 17/6 than 18/6, reaching 20% in the former case. The afternoon sessions lasted from 14.00 to 20.00. The peak terminal load (as measured by users logged in) was recorded at around 16.00 in both sessions, and then dropped off steadily. It is therefore likely that at the peak load, the terminal CPU utilisation was much higher. In the 18/6 session, the batch bottleneck coincided with the peak terminal load.

The figures for the 'number of times no PP was available' are higher for the afternoon sessions than the morning sessions. The highest morning figure was a rate of 2,260/hour on the morning of 18/6. The afternoon figures were 3,570/hour and 2,640/hour on the 17/6 and 18/6 respectively. This indicates that the overall system load was higher in the afternoon sessions than the morning sessions. However the average load appears to have been higher on the 17/6 afternoon, a session in which the terminal load was higher. In spite of this, the short jobs suffered less delay than on 18/6, recording a mean elapsed time of 46 seconds on 17/6 compared with 104 seconds on 18/6.

Table 7.13 shows the main characteristics of the short job workload for the two afternoon sessions. Comparing the characteristics for the untreated sets of jobs on 17/6 and 18/6, it can be seen that most of the characteristics are similar. However, one major difference is that whereas each short job had an average of 2 other short jobs competing with it on 17/6, this figure increased to 4.3 on the 18/6. This analysis indicates that the considerably longer short job elapsed times on 18/6, were due more to competition from other short jobs than from terminal user activity (see also 7.4.2).

Table 7.13: Characteristics of Short Job Workload
Afternoon Samples

Characteristics		17/6/74 Untreated	18/6/74 Untreated	17/6/74 Treated
Number of jobs		872	831	799
Elapsed time(secs)	m.	45.7	104.4	29.7
	s.d.	69.4	411.6	24.9
CPU time (CPU)(secs)	m.	5.0	5.3	4.8
	s.d.	5.8	5.6	5.6
Disc records trans-	m.	275	304	258
ferred	s.d.	310	357	282
No. of control cards (NCC)	m.	5.2	6.2	4.7
	s.d.	6.3	6.2	3.4
No. jobs competing (AVB)	m.	8.7	10.8	8.4
	s.d.	4.6	6.8	4.5
No. short jobs competing (AVJ)	m.	2.0	4.3	1.7
	s.d.	2.0	4.9	1.8

7.6.2 Constructing the Models

The procedure for modelling the afternoon workload was the same as that adopted for modelling the morning workload (section 7.5). Large residuals were gradually excluded, with the objective of eventually constructing satisfactory models. With the 18/6 session, this proved to be a very difficult task. With 136 observations excluded, the fit was still very poor. As described in the previous section, this session displayed highly unusual characteristics. Constructing a model for this session was abandoned because, to construct an adequate model, so many observations would have had to be excluded that it is doubtful whether the final model would have been meaningful.

In order to obtain a good fit for the 17/6 model, 73 observations were deleted. The remaining jobs constitute the treated set, and their main characteristics are displayed in table 7.13. A major difference between the afternoon treated set and the morning treated sets (table 7.5) is the larger mean elapsed time of almost 30 seconds for the afternoon set. This contrasts with a mean elapsed time of between 20 and 22 seconds for the morning sessions. Furthermore the average number of short jobs competing with a given short job was 1.7. This was higher than any of the morning sessions, where the highest figure recorded was 1.2 on 12/6 morning.

7.6.3 The Models

7.6.3.1 Introduction

Three versions of the model were constructed. They are shown in table 7.14, and are similar in form to the three versions constructed for the morning sessions. In general R^2 is lower than in the morning models and the standard error of the residuals is considerably higher.

Table 7.14: Regression Model of Short Job Workload
(17/6/74 p.m. sample)

Independent Variable	Version 1	Independent Variable	Version 2	Independent Variable	Version 3
CPU		CPU		CPU*AVT	
r.c.	2.89	r.c.	1.67	r.c.	0.07
s.e.	0.10	s.e.	0.12	s.e.	0.005
t	28.9	t	13.9	t	14.0
NCC		NCC		NCC	
r.c.	1.92	r.c.	1.95	r.c.	1.89
s.e.	0.16	s.e.	0.16	s.e.	0.16
t	12.0	t	12.2	t	11.8
AVJ		CPU*AVJ		CPU*AVJ	
r.c.	3.96	r.c.	0.70	r.c.	0.58
s.e.	0.30	s.e.	0.05	s.e.	0.05
t	13.2	t	14.0	t	11.6
Intercept	-0.08	Intercept	7.36	Intercept	7.67
R^2	0.63	R^2	0.64	R^2	0.66
s	15.1	s	14.9	s	14.6
F	456	F	481	F	504
No. of jobs	799	No. of jobs	799	No. of jobs	799

For key to abbreviations refer to table 7.7

The residual plot in figure 7.9 shows that the biggest residuals in the afternoon model are larger than their counterparts in the morning models (figures 7.5, 7.6 and 7.8). Apart from this, the afternoon residual plot shows no major differences from the morning residual plots.

7.6.3.2 The First Version (CPU, NCC, CPU*AVJ)

In the first version of the model (CPU, NCC, AVJ), the values of the coefficients of the three variables are all larger than any of their counterparts in the morning sessions (table 7.8). This is partly explained by the fact that the elapsed times in the afternoon are on average 50% higher than in the morning. Therefore, for the same set of independent variables, the higher predicted elapsed times must either come from larger regression coefficients or from a larger intercept. In this version of the model, the intercept is small, and the coefficients of the CPU and NCC variables are larger than in the morning models.

7.6.3.3 The Second Version (CPU, NCC, CPU*AVJ)

In the second version of the model (CPU, NCC, CPU * AVJ), the intercept is 7.36 which is much larger than in the first version (-0.08) and is also larger than any of the corresponding morning models (table 7.9). The CPU coefficient is relatively close to the morning models, whereas the NCC coefficient is larger.

7.6.3.4 The Third Version (CPU * AVT, NCC, CPU * AVJ)

In the third version of the afternoon model (CPU * AVT, NCC, CPU * AVJ) the intercept is 7.67, which is about the same size as in the second version, but larger

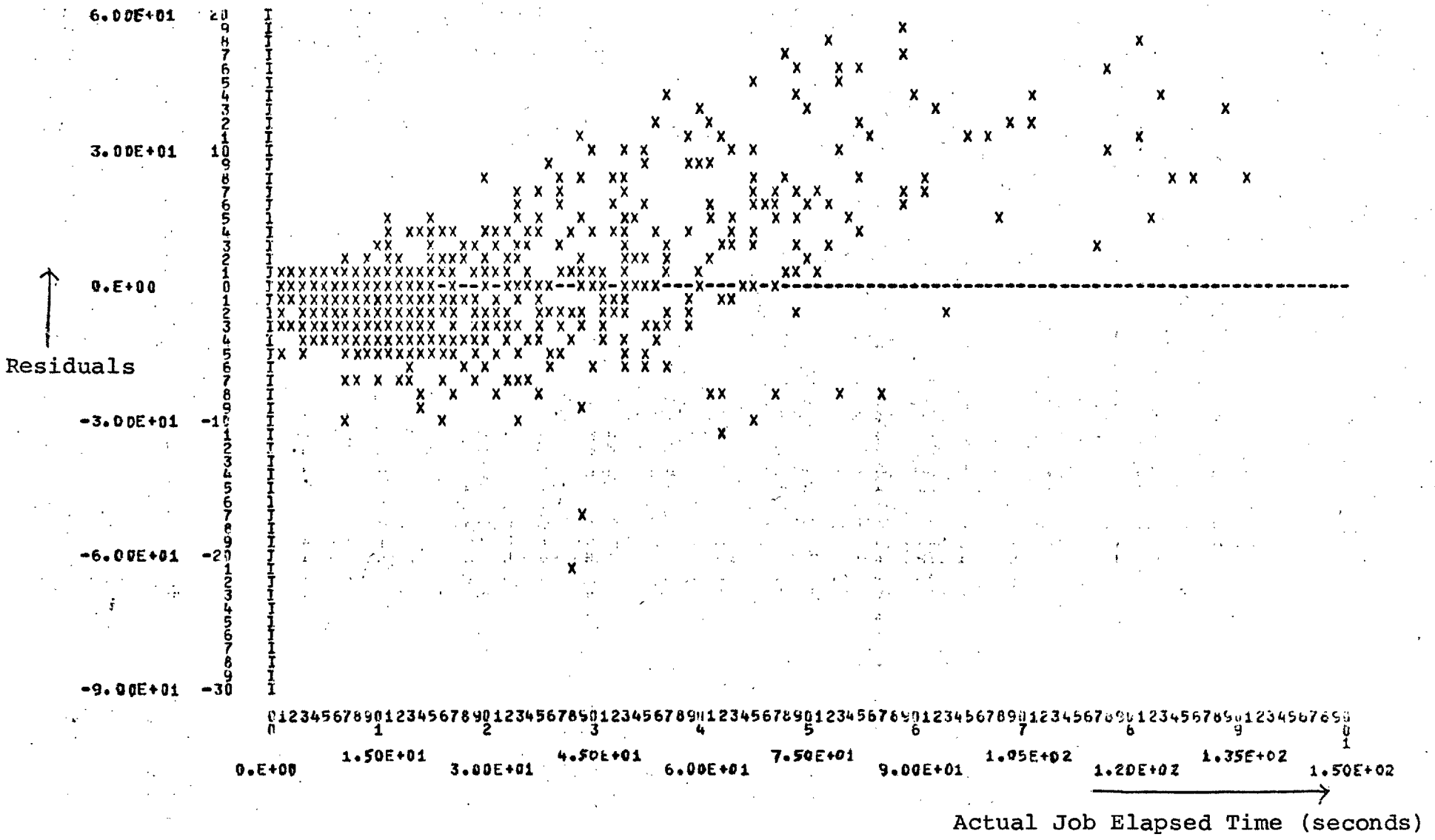


Figure 7.9: Residual Plot - Model of Short Job Workload (17/6/74 p.m. session)

than in corresponding morning models (table 7.10). R^2 is marginally larger than for the first two versions. NCC is again larger than for the morning models.

7.6.4 Comparison of Models

The second and third versions are not very satisfactory, because of their large intercepts. In the first version, which has a small intercept, the CPU and NCC coefficients are larger than in the morning models. The AVJ coefficient is however about the same value. This means that the increased delay that the afternoon short jobs experienced is being attributed to the CPU and NCC coefficients. In the second and third versions it is being attributed to the intercept.

When the load is high, jobs are delayed because they are rolled out of CM. Jobs with larger resource demands (e.g. CPU time) are likely to suffer comparatively greater delays. Hence, in the absence of a good measure of load on the system, the coefficient of the CPU variable, the independent variable most highly correlated with elapsed time, is larger for sessions with heavier loads.

As pointed out in section 7.5.4.2, the coefficient of NCC (the number of job steps) may be considered as a measure of the system overhead to initiate a job step. In the afternoon session, the load was higher than in the morning sessions. It is likely that the system overhead to initiate each job step would also on average be higher. The larger NCC coefficient reflects this increased overhead.

In conclusion, the main reason for the difference between the afternoon model and the morning models, is the heavier system load (both batch and timesharing) in the afternoon session.

7.7 Regression Modelling of the Batch Workload in the Absence of the Time Sharing Load

7.7.1 Introduction

A new machine, the CYBER 7314, was installed at the Imperial College Computer Centre during the summer of 1974. Between August 1974 and June 1975, while software was being developed locally to allow a shared permanent file base, the CYBER 7314 and 6400 operated independently of each other. The CYBER supported the bulk of the batch workload, including the cafeteria service, the local batch workload and the remote batch workload. The 6400 supported the timesharing workload and the batch jobs submitted by timesharing users.

The operating system run on both systems was Kronos 2.1. The main differences between Kronos 2.1 and the previous version, Kronos 2.0, are the additional user facilities provided and the new peripherals supported by Kronos 2.1. Resource management is similar to Kronos 2.0.

The 6400 and CYBER 7314 are architecturally similar apart from the compare/move unit (section 4.3). The configuration on both machines was also enhanced with the addition of faster disc units.

The fact that, for a temporary period of time, the CYBER was to support an entirely batch workload, provided an excellent opportunity for the batch workload to be modelled in the absence of the timesharing workload. A complete comparison with the previous analysis would not however be possible because of differences in the architecture, machine configuration and operating system, between the 1974 evaluation on the 6400 and the 1975 evaluation on the CYBER.

7.7.2 Characteristics of the Workload

The workload data for the evaluation of the entirely batch workload on the CYBER system, was gathered during two periods in January and April 1975. Four sessions, three morning and one afternoon, were used for the evaluation. The morning sessions were on the 27th January, 28th and 30th April. The afternoon session was on the 30th January. Both the Account and System Dayfiles were collected for each session.

The main overall characteristics of the four sessions are displayed in Table 7.15, and the main characteristics of the short job workload are displayed in Table 7.16. The morning sessions were of approximately the same length, between 3¼ and 3½ hours. The afternoon period was the first two hours of a longer session. During this period, roughly the same number of short jobs were processed as in the morning sessions. Thus the afternoon session experienced a considerably higher average short job load. Table 7.16 shows that whereas the average number of short jobs in competition with a given job varied between 0.42 and 0.67 in the morning sessions, it was 1.18 in the afternoon session. Table 7.15 shows that, as a result of the greater load, the short job CPU utilisation was 16.4% in the afternoon, whereas in the three mornings it varied from 8.5% to 11.7%.

In the four sessions, the short job workload accounted for between 75% and 85% of all jobs processed. However, the long job CPU utilisation was over four times as large as the short job CPU utilisation, except for the afternoon session where it was only twice as large. The long job CPU utilisation on 30/1/75 is probably an underestimate of the real figure, as it does not include any jobs which started before the end of the monitored period, and finished after it.

Table 7.15: Characteristics of the 1975 Batch Workload

	27/1/75	30/1/75	28/4/75	30/4/75
Start of Monitored Period	9.37.30	14.03.09	9.31.30	9.36.59
End of Monitored Period	12.52.13	16.04.32	12.59.59	12.50.46
All Jobs				
Number of jobs	415	454	355	467
Total CPU time (secs)	6875	3841	6231	7810
Mean CPU time/job (secs)	19.4	8.5	17.6	16.7
Batch CPU utilisation	59%	53%*	50%	67%
Short Jobs				
Number of Jobs	354(85%)	364(80%)	278(78%)	352(75%)
Total CPU time (secs)	1315	1181	1061	1360
Mean CPU time (secs)	3.72	3.45	3.82	3.86
Mean elapsed time (secs)	19.4	22.9	21.9	20.0
Short job CPU utilisation	11.3%	16.4%	8.5%	11.7%
Long Jobs				
Number of Jobs	61	90	77	115
Total CPU time (secs)	5560	2660	5170	6450
Mean CPU time (secs)	91.3	29.5	67.2	56.0
Mean elapsed time (secs)	961	445	279	443
Long job CPU utilisation	48%	36.8%	41.3%	55.5%

* Approximate figure

Table 7.16: Characteristics of the 1975 Short Job Workload (Four Samples Untreated)

Characteristic		27/1/75 a.m.	28/4/75 a.m.	30/4/75 a.m.	30/1/75 p.m.
Number of jobs		354	278	352	364
Elapsed time(secs)	m.	19.4	21.9	20.0	22.9
	s.d.	18.1	26.1	17.1	33.9
CPU time(secs) (CPU)	m.	3.72	3.82	3.86	3.45
	s.d.	4.90	4.62	4.95	4.94
Disc records transferred (DPRU)	m.	239	384	234	234
	s.d.	462	780	216	379
No.control cards (NCC)	m.	3.06	5.08	3.71	2.99
	s.d.	3.00	7.15	4.24	3.79
Average CM(kilowords)	m.	13.4	14.6	15.4	14.2
	s.d.	8.5	8.8	7.6	7.8
Average short job load (AVJ)	m.	0.67	0.42	0.49	1.18
	s.d.	0.89	0.67	0.57	1.05
Average total job load (AVB)	m.	6.10	2.28	4.83	7.57
	s.d.	2.38	1.61	2.62	2.20

Key: m. : mean
s.d. : standard deviation

Comparing the short job workload characteristics (table 7.16) with the 6400 set (table 7.4), shows that the mean CPU time was noticeably lower on the CYBER (mean = 3.6 secs) than the 6400 (mean = 4.4 secs). The difference may reflect a genuine change in the workload, or may be due to the effect of the additional character handling instructions provided by the CYBER's compare/move unit and used by some of the system software. It is probably a combination of both. The mean short job elapsed times are considerably lower on the CYBER. The main reason for this is the absence of the timesharing load.

7.7.3 The Initial Models

The initial models for the four sessions are shown in Table 7.17. The Forward Selection Regression procedure was again used to select the independent variables to be included in the model. As observed before, there is a wide variation in the independent variables selected for each model.

Comparing the initial models on the CYBER with those on the 6400 (table 7.6), one major difference is apparent. In the initial models on the 6400, R^2 was not at all satisfactory. In the initial models of the three morning sessions on CYBER, R^2 is over 0.67, which is considered satisfactory. However R^2 is much lower for the 30/1/75 model. An analysis of residuals for this model showed that two jobs with large elapsed times had very large positive residuals. Inspection of the Dayfiles revealed that both these jobs were run under the same user number. In each case, the job had hung at a control point and eventually been dropped by an operator. When these jobs are excluded, the fit improves considerably.

This shows that now the timesharing load has been removed, much better models of the short job workload can be constructed.

Table 7.17: Initial Models of the Short Job Workload

27/1/75 a.m. sample - 354 observations

$$Y = 5.4 + 0.23CPU*AVB + 0.01DPRU + 1.51NCC*AVJ + 1.50CPU - 0.19WH$$

$$R^2 = 0.79 \quad s = 8.3 \quad F = 264$$

28/4/75 a.m. sample - 278 observations

$$Y = 6.37 + 0.01DPRU*AVJ + 2.65CPU + 1.49NCC*AVJ$$

$$R^2 = 0.80 \quad s = 11.7 \quad F = 278$$

30/4/75 a.m. sample - 352 observations

$$Y = 3.20 + 2.39CPU + 1.07NCC*AVJ + 0.76AVB + 0.55NCC$$

$$R^2 = 0.67 \quad s = 9.9 \quad F = 177$$

30/1/75 p.m. sample - 364 observations

$$Y = 9.91 + 0.55CPU*AVB + 0.85NCC*AVJ + 0.00001DPRU^2 - 0.29WH$$

$$R^2 = 0.30 \quad s = 28.7 \quad F = 37.5$$

For key to abbreviations refer to table 7.6

7.7.4 Analysis of Residuals

The residual plots are similar in character to the 1974 plots, although not so accentuated. An example is shown in figure 7.10 for the 30/4 session. There are a large number of small negative residuals and fewer large positive residuals. The large positive residuals occur for jobs with large elapsed times.

An analysis of residuals showed that there were some jobs in each session which displayed unusual characteristics. The characteristics of these jobs were one or more of the following:

(i) Large job elapsed times and large positive residuals.

(ii) An uncharacteristically large number of job steps.

(iii) An uncharacteristically large number of disc physical records transferred.

An examination of the System Dayfile showed that although these uncharacteristic jobs were relatively few in number, they were sometimes run a number of times in each session. Some of the uncharacteristic jobs always had long elapsed times. Others only recorded long elapsed times under certain conditions, such as if other uncharacteristic jobs were running with them or if the short job load was comparatively high. Ordinary jobs, which were run at the same time, were sometimes delayed by these jobs.

The uncharacteristic jobs distort the models (see 6.5.4). The most frequently observed ones were:

(a) Jobs which use the microfilm feature.

(b) Jobs which use the facility for editing a user program library.

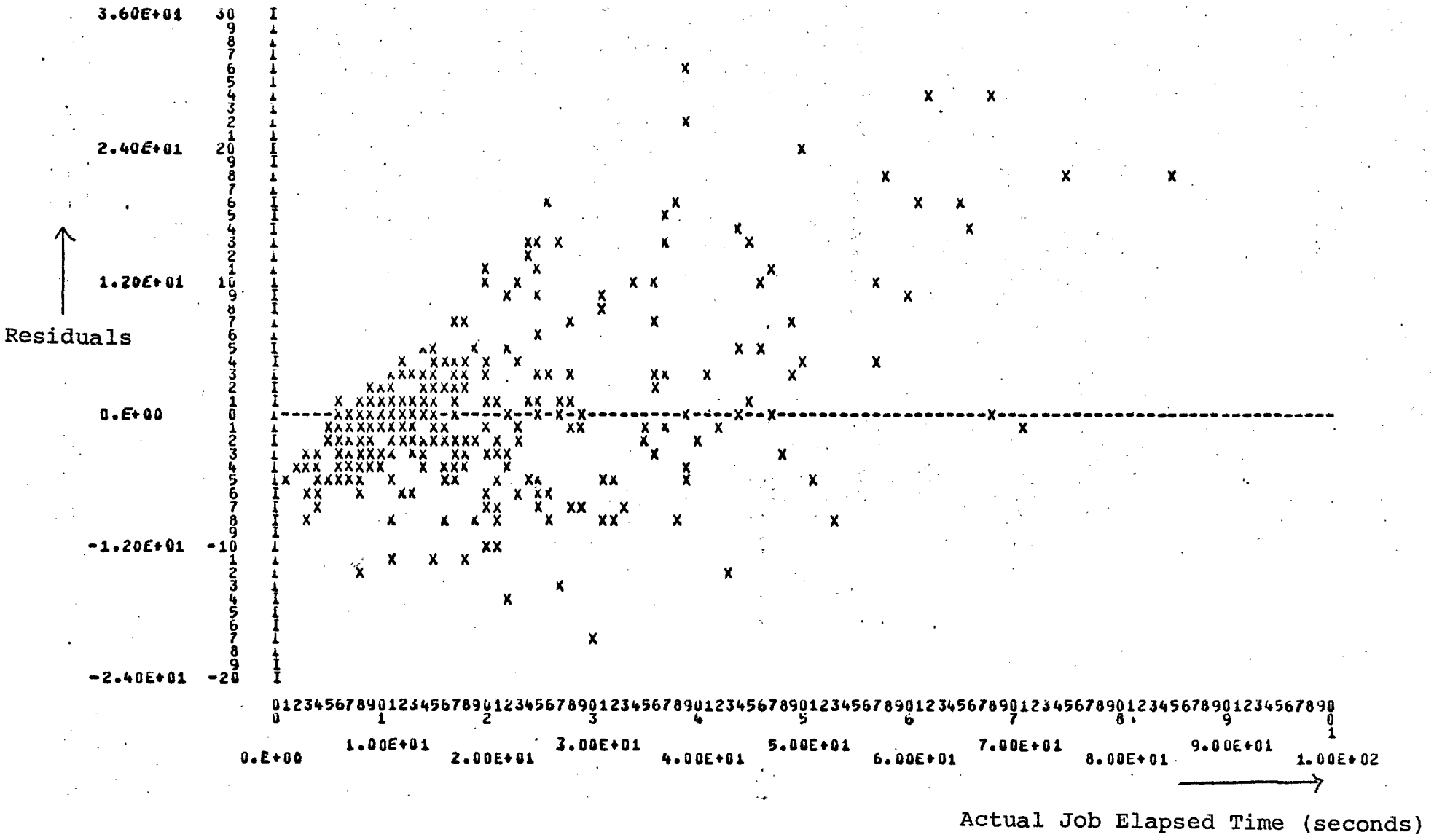


Figure 7.10: Residual Plot - Model of Short Job Workload (30/4/75 session)

(c) Jobs which test a particular mathematical program library. This was the same type of job which had the adverse effect on performance discussed in section 7.4.2.

(d) Jobs which use the utility for querying which magnetic tapes their permanent files had been archived to.

All these jobs are likely to be highly I/O bound. Many of these jobs (e.g. the first three types above) are jobs one might not normally expect to be run in the short job category.

It was decided that these uncharacteristic jobs, which constitute a small percentage of the short job workload, should be excluded from the analysis. The characteristics of the short job workload, after excluding these jobs, are shown in table 7.18.

7.7.5 Models of the Short Job Workload

The models for the four sessions, constructed after excluding uncharacteristic observations, are shown in table 7.19. The three independent variables, CPU (job CPU time), NCC (number of job steps) and AVJ (short job load) were forced in as before.

Table 7.19 shows that R^2 is between 0.7 and 0.75 for the four models. The regression coefficients for the CPU variable are in close agreement, as are the coefficients of NCC. However, there is a greater variability in the coefficient of the AVJ variable.

Comparing these models with the 1974 models (table 7.8), it can be seen that whereas the coefficients

Table 7.18: Characteristics of the 1975 Short Job Workload (Four Samples Treated)

Characteristics		27/1/75 a.m.	28/4/75 a.m.	30/4/75 a.m.	30/1/75 p.m.
Number of jobs		333	259	345	354
Elapsed time (secs)	m.	18.1	16.8	18.9	19.6
	s.d.	14.1	15.1	14.8	15.4
CPU time (secs) (CPU)	m.	3.61	3.50	3.75	3.31
	s.d.	4.74	4.27	4.81	4.81
Disc records transferred (DPRU)	m.	206	294	234	215
	s.d.	162	455	216	214
No. control cards (NCC)	m.	3.04	4.43	3.70	2.86
	s.d.	2.78	5.14	4.24	3.54
Average CM(Kilowords)	m.	13.7	14.8	15.5	14.2
	s.d.	8.6	8.7	7.5	7.8
Average short job load (AVJ)	m.	0.57	0.35	0.49	1.17
	s.d.	0.65	0.59	0.57	1.06
Average total job load	m.	5.97	2.21	4.80	7.55
	s.d.	2.23	1.54	2.60	2.21
No. jobs excluded		19	19	7	10

Table 7.19: Regression Models of the Short Job Workload (outliers excluded)

Independent Variable		27/1/75 a.m.	28/4/75 a.m.	30/4/75 a.m.	30/1/75 p.m.	combined a.m.	combined all
Equation Number		1	2	3	4	5	6
CPU	r.c.	2.25	2.22	2.21	2.28	2.24	2.25
	s.e.	0.09	0.12	0.09	0.09	0.06	0.05
	t	25.0	18.8	24.8	25.1	40.1	47.3
NCC	r.c.	1.16	1.05	1.05	1.14	1.05	1.08
	s.e.	0.15	0.10	0.10	0.12	0.06	0.06
	t	7.6	10.5	10.5	9.3	16.8	19.3
AVJ	r.c.	5.47	7.17	5.38	4.92	6.01	5.32
	s.e.	0.65	0.82	0.72	0.41	0.41	0.26
	t	8.5	8.8	7.4	12.1	14.6	20.1
Intercept		3.38	1.92	4.02	3.06	3.18	3.25
Adjusted Intercept (Equation 5)		3.44	2.25	3.64			
Adjusted Intercept (Equation 6)		3.72	2.33	3.83	2.91		
R^2		0.71	0.75	0.73	0.73	0.73	0.73
s		7.7	7.7	7.7	8.1	7.6	7.8
F		266	249	311	318	830	1145
No. of jobs		333	259	345	354	937	1291

For key to abbreviations refer to table 7.7

of the CPU variable are quite close for the 74 and 75 models, the coefficients of NCC are smaller in 75. As pointed out in section 7.5.4.2, the coefficient of NCC may be viewed as a measure of the system overhead for job step initiation. With no interactive load, it is not surprising that the overhead is less.

Table 7.19 also shows the model constructed using data pooled from all three morning sessions, and the model constructed using data pooled from all four sessions.

7.8 Validation of the Models

7.8.1 Comparison of Slopes

Table 7.19 shows that the regression coefficients of the independent variables are close. However, are they close enough for the models to be considered statistically consistent? The method used for testing this is an extension of the method described in reference S13. The method tests whether there is any significant difference in the slopes (in four dimensions) of the models. This is done by comparing the residual sum of squares of the models of the individual sessions with that of the pooled model.

The method was used first to compare the regression coefficients of the three morning sessions, as shown in table 7.20. The residual sum of squares for the three separate models and their respective degrees of freedom are derived first (lines 1, 2 and 3), and summed (line 4). The residual mean square of 58.7 is derived (line 4), which represents the mean square when the regression models for the individual sessions are fitted to each set of data respectively.

The pooled model is then constructed for all the data (equation 5 in table 7.19). This model is applied

Table 7.20: Comparison of Morning Models

	Model	No. of observations	d.f.	Analysis of Residuals		
				d.f.	RSS	MSR
1	27/1/75	333	332	329	19264	58.5
2	28/4/75	259	258	255	14972	58.6
3	30/4/75	345	344	341	20160	59.0
4				925	54396	58.7
5	Pooled applied to 27/1	333			19335	
6	Pooled applied to 28/4	259			15093	
7	Pooled applied to 30/4	345			20209	
8		937	934	931	54637	58.7
9	Difference between slopes			6	241	40.2
10	Pooled applied to all data	937	936	933	54957	
11	Difference between intercepts			2	320	160

Comparison of slopes: $F = 40.2/58.7 = 0.69$ (d.f.=6,925)
Difference not significant at 5% level

Comparison of Intercepts: $F = 160/58.7 = 2.72$ (d.f=3,931)
Difference not significant at $2\frac{1}{2}$% level

Key: d.f.: degrees of freedom RSS = residual sum of squares
MSR : Residual Mean square = RSS/d.f.

to each set of data after adjusting for the intercept. For each set of data, the adjusted intercept bo' is given by

$$bo' = \bar{Y} - b_1 \bar{T} - b_2 \bar{K} - b_3 \bar{N}$$

where b_1, b_2 and b_3 are the regression coefficients of the pooled model. $\bar{Y}$, $\bar{T}$, $\bar{K}$, $\bar{N}$ are the mean job elapsed time, CPU time, number of job steps and short job load respectively. This is to ensure that the regression equation passes through the centroid of the sample.

After applying the pooled model to each set of data, the residual sum of squares is derived for each set (lines 5, 6 and 7) and summed (line 8). The difference in the residual sum of squares

54637 (line 8) - 54396 (line 4) = 241 (line 9)

with six degrees of freedom, measures the contribution of the differences between the regression coefficients to the sum of squares of residuals. The corresponding mean square, 40.2 in line 9, is compared with the mean square in line 4, using the F-test, to test if there is any significant difference when the slopes of the models of individual sessions are compared with the slope of the pooled model.

$$F = \frac{40.2}{58.7} = 0.69 \text{ (degrees of freedom = 6;925)}$$

0.69 is less than the F value at the 5% level of significance (2.10), supporting the assumption that the slopes do not differ.

7.8.2 A Non-Parametric Significance Test

The F-test makes the assumption that the residuals are normally distributed, an assumption which, as the

Kolmogorov-Smirnov one-sample test shows, is not strictly valid. As a further check, a non-parameteric test, the Mann-Whitney U-Test (S8) was also carried out. Non-parametric tests do not make any assumptions about the distribution of the residuals. The residuals squared, derived by applying the pooled model with adjusted intercept to each set of data, are compared by the test. The test is applied to two sets of residuals at a time, making three tests in all, and the results are displayed in table 7.22a. The assumption that each two sets of residuals squared are drawn from the same population is tested. The result of the test is the probability of the assumption being true.

Table 7.22a shows that the probability (P_1 column) is 5% or more for the three cases, supporting the assumption that the slopes do not differ. Hence, the result of the Mann-Whitney test supports the result of the F-test. This means that the assumption that the slopes do not differ, and hence that the regression coefficients are consistent, may be accepted.

7.8.3 Comparison of Intercepts

Once it has been shown that the regression coefficients of the models are consistent, it is then permissible to test whether the adjusted intercepts are consistent. This is done by comparing the residual sum of squares (RSS) of the pooled model applied to all the data (equation 5 in table 7.19) with the RSS of the pooled model applied to each set of data after adjusting the intercept for each set (table 7.20). The latter RSS, 54637 in line 8 is subtracted from the former RSS, 54957 in line 10, giving 320 (line 11) with 2 degrees of freedom, which measures the contribution of the differences between the intercepts to the residual sum of squares. The corresponding mean square, 160 in line 11, is compared with the mean square in line 8, 58.7, using the F-test to test if there is any

significant difference in the intercepts. The F-test shows that although there is a significant difference at the 5% level, the difference is not significant at the $2\frac{1}{2}$% level. Thus the assumption that there is no difference in the intercepts is accepted with less confidence than the assumption that there is no difference in the slopes.

As a separate check, the Mann-Whitney U-test was used to compare the residuals squared obtained from the pooled model, applied to each set of data (equation 5 in table 7.19). The test (P2 column in table 7.22a) shows that at the 5% level, there is a significant difference between the 27/1 and 28/4 sessions but no significant difference between the 27/1 and 30/4 sessions, and the 28/4 and 30/4 sessions respectively. When the pooled model with adjusted intercept was applied to the 27/1 and 28/4 sets of data there was no significant difference at the 5% level. Consequently the reason for the change must be due to the difference in the two intercepts.

These results indicate that there probably is a slight difference in the intercepts. However as the difference in the intercepts is small compared with the standard error of the residuals (table 7.19), the difference is not considered serious. The difference in the intercepts may be due to a slight difference in the environments on the days in question. Unfortunately, ICCC records only indicate major changes in the environment, e.g. system up or down, so this cannot be checked.

7.8.4 Comparison with Afternoon Model

The validation of the morning models was described in the previous three subsections. Because of the larger short job load in the afternoon session (30/1/75), some difference might be expected in the afternoon model. The same analysis as before was carried out to compare the regression models of all four sessions. The models of the

individual sessions are compared with the pooled model (equation 6 in table 7.19). First the slopes are compared and then the intercepts.

The results of the F-test are shown in table 7.21 and of the Mann-Whitney U-test in table 7.22b. First the slopes are compared using the F-test. Table 7.21 shows that there is no significant difference at the 5% level.

Next, the slopes are compared using the Mann-Whitney test. The 30/1 set of residuals squared is compared with each of the three morning sets of residuals squared in table 7.22b (P_1 column). The test indicates that with the 28/4 and 30/4 residuals, there is no significant difference at the 5% level. However, with the 27/1 residuals, the test suggests that there is a significant difference at the 5% level, but no significant difference at the 1% level. Comparing the two sets of regression coefficients for the 27/1 and 30/1 models in table 7.19, it can be seen that they agree closely. It therefore seems reasonable to accept the assumption that there is no significant difference between them.

The adjusted intercepts of the four models are compared (table 7.21), next. The F-test indicates that there is no significant difference between the intercepts at the 5% level.

Finally, the Mann-Whitney U-test is used to compare the residuals squared obtained by applying the pooled model to each set of data (equation 6 in table 7.19). This test (table 7.22b) indicates that there is a significant difference between the 27/1 and 30/1 sessions, but no significant difference between the 30/1 session and the 28/4 and 30/4 sessions respectively.

The results indicate that there is no reason to believe that there is a significant difference between the afternoon model and the morning models. There is, however,

Table 7.21: Comparison of Four Models

	Model	No.of observations	d.f.	Analysis of Residuals d.f.	RSS	MSR
1	27/1/75	333	332	329	19264	58.5
2	28/4/75	259	258	255	14972	58.6
3	30/4/75	345	344	341	20160	59.0
4	30/1/75	354	353	350	22595	64.5
5				1275	76991	60.3
6	Pooled Applied to 27/1	333			19286	
7	Pooled Applied to 28/4	259			15285	
8	Pooled Applied to 30/4	345			20178	
9	Pooled Applied to 30/1	354			22698	
10		1291	1287	1284	77447	60.2
11	Difference between slopes			9	456	50.7
12	Pooled Applied to all data	1291	1290	1287	77897	
13	Difference between intercepts			3	450	150

Comparison of slopes: $F = 50.7/60.3 = 0.84$ (d.f. = 9,1275)
Difference not significant at 5% level

Comparison of Intercepts: $F = 150/60.2 = 2.49$ (d.f. = 3,1284)
Difference not significant at 5% level

Table 7.22: Comparison of Models using Mann-Whitney U-test

Table 7.22a: Comparison of three morning models

Pairwise Comparison		P_1	P_2
1st Model	2nd Model		
27/1/75	28/4/75	0.067	0.006
27/1/75	30/4/75	0.050	0.090
28/4/75	30/4/75	0.488	0.115

Key: P : probability that there is no difference between the two models

P_1: Pooled model with intercepts adjusted

P_2: Pooled model without intercept adjusted

Table 7.22b: Comparison of four models

1st Model	2nd Model	P_1	P_2
27/1/75	30/1/75	0.018	0.003
28/4/75	30/1/75	0.436	0.359
30/4/75	30/1/75	0.337	0.104

Key: see table 7.22a.

some indication that the environment on 27/1 may have been slightly different from the other days.

7.8.5 The Workload Model

The validation of the models has shown that the regression coefficients of the four models are consistent, although there is a greater variation in the intercepts. It is therefore justifiable to use the model pooled from all four sessions as a satisfactory regression model of the short job workload. This model is called the Workload Model and is represented by equation 6 in table 7.19.

7.9 Regression Models with no Short Job Competition

7.9.1 Introduction

In the regression models described in sections 7.7 and 7.8, the regression coefficients with the largest variation were the coefficients of AVJ, the measure of the short job load. In an attempt to determine what the effect of this independent variable is, a subset of the workload was modelled. A substantial number of the jobs executed in each morning session, experienced no competition from other short jobs. This subset of jobs was isolated for each session and its characteristics are shown in table 7.23.

Comparison of the workload characteristics of this subset (table 7.23) with the total set (table 7.16) reveals a number of differences. Firstly, the mean elapsed time is much lower when there is no competition from other short jobs. Secondly, the mean CPU time is substantially lower for each of the three samples. The mean number of job steps is also lower. Thus, there appears to be a tendency for the jobs in this subset to have lower resource demands. Jobs with larger resource demands experience longer elapsed times and so are more likely to experience competition from other short jobs. This means that the subset is not a representative one. Nevertheless, it was felt that modelling the subset could provide further insight into the behaviour of the short job workload.

Table 7.23: Characteristics of Subset of Short Job Workload

Characteristic		27/1/75 a.m.	28/4/75 a.m.	30/4/75 a.m.
Number of jobs		124	139	129
Elapsed time (secs)	m.	10.8	10.9	11.3
	s.d.	6.8	7.2	7.2
CPU time (secs)	m.	2.44	2.77	2.28
(CPU)	s.d.	3.39	3.32	3.01
Disc records transferred	m.	187	201	193
(DPRU)	s.d.	143	157	123
No. control cards	m.	2.40	4.00	3.09
(NCC)	s.d.	1.99	4.33	3.99
Average CM (kilowords)	m.	13.9	14.7	15.6
	s.d.	8.4	9.2	7.5
Average total job load	m.	4.65	1.53	4.50
(AVB)	s.d.	1.89	1.24	2.68
No. jobs excluded		1	7	3

7.9.2 The Models

Regression models were constructed for the subset of each session and for the pooled data. A few uncharacteristic observations were excluded, for the same reasons as given in section 7.7.4. The CPU and NCC independent variables were forced in. The models are displayed in table 7.24. A number of interesting points arise from this analysis:

(a) The amount of variation explained by the models (R^2) is satisfactory, 0.66 or above.

(b) The regression coefficients for CPU and NCC are of substantially lower value than in the models of the total short job workload. The reasons for this are discussed in 7.9.3.

(c) The standard error of the residual(s) is considerably lower than before, ranging from 3.3 to 4.2. This compares with the models of the total job workload, where s varied from 7.7 to 8.1 (table 7.19).

The substantially smaller mean elapsed times and standard error for these subsets is due to the fact that the short jobs in this subset should never have been rolled out. This is because:

(i) Each job was always the highest priority user job executing.

(ii) Each job experienced no competition from other short jobs.

(d) An analysis of residuals reveals that the tendency so far for the positive residuals to be substantially fewer in number and larger in magnitude, has now been eliminated. An example is shown in figure 7.11

Table 7.24: Regression Models of Subset of Short Job Workload (Two Independent Variables)

Independent Variable		27/1/75 a.m.	28/4/75 a.m.	30/4/75 a.m.	Pooled
CPU	r.c.	1.70	1.53	1.43	1.55
	s.e.	0.09	0.11	0.12	0.06
	t	19.2	14.1	11.7	24.6
NCC	r.c.	0.56	0.48	0.75	0.56
	s.e.	0.15	0.08	0.10	0.06
	t	3.7	5.7	7.9	10.0
Intercept		5.32	4.72	5.75	5.34
Adjusted Intercept		5.69	4.34	6.08	
R^2		0.77	0.68	0.66	0.68
s		3.29	4.1	4.2	4.0
F		205	142	124	418
No.of jobs		124	139	129	392

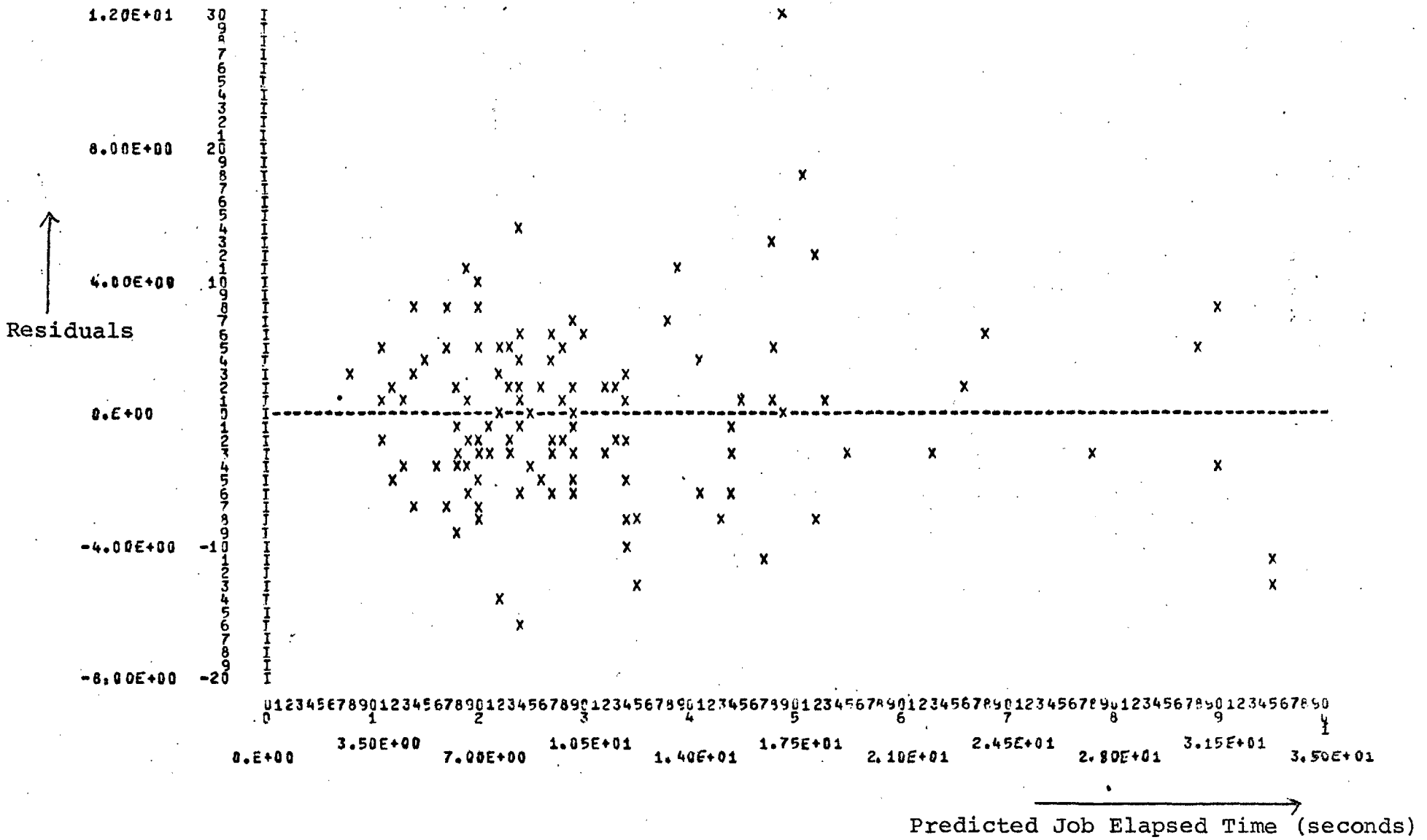


Figure 7.11: Residual Plot - Model of Subset with no Short Job Competition (27/1/75 session)

of the residual plot against predicted elapsed time for the 27/1 model. Apart from one outlier, the magnitude of the positive residuals is not noticeably different from the magnitude of the negative residuals.

The reason for this is again likely to be due to the fact that these short jobs should never have been rolled out.

(e) The Kolmogorov-Smirnov, one-sample test (S8) was applied to the residuals of each model to test the assumption that the residuals are normally distributed. The test indicates the probability of the assumption that the residuals are normally distributed being true. The probabilities of 0.29 for 27/1 and 0.11 for 30/4 are higher than the 5% significance level. The probability of 0.04 for 28/4 is just below the 5% level. However, it is sufficiently close to justify accepting the normality assumption for all three sets of residuals.

This is the first time in the regression analysis of the Kronos system that the normality assumption has been accepted. The reason is likely to be the elimination of the distortion in the residual plot (see d). The F and t tests may now be used with full confidence.

(f) The intercepts for the models are larger than those for the models of the total short job workload. The intercept decreases in value when more variables are introduced into the model. When AVB, the number of jobs concurrently in execution, is introduced, R^2 is increased in value and the intercept is reduced, as shown in table 7.25. AVB is not significantly correlated with CPU and NCC, and its introduction does not much affect the CPU and NCC coefficients. However, table 7.25 shows that there is a greater variability in the coefficient of AVB

Table 7.25: Regression Models of Subset of Short Job Workload (Three Independent Variables)

Independent Variable		27/1/75 a.m.	28/4/75 a.m.	30/4/75 a.m.	Pooled
CPU	r.c.	1.73	1.55	1.49	1.59
	s.e.	0.09	0.10	0.11	0.06
	t	19.9	15.1	13.1	27.0
NCC	r.c.	0.58	0.54	0.74	0.61
	s.e.	0.15	0.08	0.09	0.05
	t	3.9	6.7	8.4	11.6
AVB	r.c.	0.42	1.15	0.61	0.60
	s.e.	0.15	0.27	0.13	0.08
	t	2.7	4.2	4.7	7.8
Intercept		3.25	2.67	2.90	2.98
Adjusted Intercept		2.68	3.11	3.13	
R^2		0.79	0.71	0.72	0.73
s		3.2	3.9	3.9	3.7
F		146	112	104	341
No. of jobs		124	139	129	392

in the three models than in the coefficients of CPU and NCC. As pointed out previously, AVB is not a good measure of load, because it does not distinguish between jobs in CM and jobs rolled out, which can be a substantial number. Thus the mean value of AVB can vary considerably from session to session as shown in table 7.23.

7.9.3 The Regression Coefficients

It was pointed out in section 7.6.4, that as the average short job load AVJ, is only an approximate measure of the load on the system, it is likely that not all the delay experienced by a short job is accounted for by the AVJ term of the model. Some of the delay may be attributed to the CPU term instead. For very high loads, jobs with larger resource demands are more likely to be rolled out and for longer periods. This delay may then be attributed to the CPU term of the model, and in particular in the form of a larger coefficient of CPU.

None of the short jobs in this subset should ever have been rolled out, since they should always have been the highest priority user jobs running. Hence, for these very light loads, the coefficient of the CPU variable is lower in value.

It was also pointed out in section 7.5.4.2 that the coefficient of NCC (number of job steps) may be considered as a measure of the system overhead to initiate a job step. For very light loads, this overhead is likely to be lower, and consequently this is reflected in the smaller NCC coefficient.

7.9.4 Validation of the Models

Attempts were made to validate the models, (CPU, NCC) by first comparing the slopes and then the intercepts,

as described in 7.8. The results are shown in table 7.26. The F-test supports the assumption that the slopes do not differ, at the 5% significance level.

It was shown in 7.9.2 that the residuals of the individual models are normally distributed. The normality assumption was also tested for the residuals of the pooled model applied to each of the individual sets of data. The residuals for 27/1 and 30/4 satisfy the normality assumption at the 20% level, while the residuals for 28/4 satisfy it at the 1% level. Consequently, the results of the F-test may be accepted with confidence. The assumption that there is no significant difference in the slopes, and hence the regression coefficients, is accepted.

When the intercepts of the three models are compared (table 7.26), however, the F-test shows that there is a significant difference in the adjusted intercepts.

A second analysis was carried out to determine if the results varied with the introduction of the AVB variable (table 7.27). In the models with AVB (table 7.25), the coefficients of AVB vary considerably, but the intercepts are smaller. The F-test rejects the assumption that there is no difference in the slopes at the 5% level, but accepts it at the 2½% level. Thus the introduction of AVB into the models has resulted in a decrease in the confidence with which the assumption is maintained. However, comparing the adjusted intercepts, the F-test indicates that there is now no significant difference in the intercepts.

These results indicate that it is the AVB variable which is the cause of most uncertainty in the models. If a better measure of system load were available, such as the mean level of multiprogramming, then it is felt that this uncertainty could be reduced considerably.

Table 7.26: Comparison of Models of no Short Job Competition (Two Independent Variables)

	Model	No.of observations	d.f.	Analysis of Residuals d.f.	RSS	MSR
1	27/1/75	124	123	121	1306	10.8
2	28/4/75	139	138	136	2324	17.1
3	30/4/75	129	128	126	2228	17.7
4				383	5858	15.3
5	Pooled Applied to 27/1	124			1336	
6	Pooled Applied to 28/4	139			2342	
7	Pooled Applied to 30/4	129			2306	
8		392	389	387	5984	15.5
9	Difference between slopes			4	126	31.5
10	Pooled Applied to all data	392	391	389	6207	
11	Difference between intercepts			2	223	111.5

Comparison of slopes: $F = 31.5/15.3 = 2.06$ (d.f.= 4,383)
Difference not significant at 5% level

Comparison of Intercepts: $F = 111.5/15.5 = 7.2$ (d.f. = 2,387)
Difference is significant

For key refer to table 7.20

Table 7.27: Comparison of Models of no Short Job Competition (Three Independent Variables)

Model	No. of observations	d.f.	Analysis of Residuals d.f.	RSS	MSR
1 27/1/75	124	123	120	1231	10.3
2 28/4/75	139	138	135	2054	15.2
3 30/4/75	129	128	125	1886	15.1
4			380	5171	13.6
5 Pooled Applied to 27/1	124			1280	
6 Pooled Applied to 28/4	139			2150	
7 Pooled Applied to 30/4	129			1926	
8	392	389	386	5356	13.9
9 Difference between slopes			6	185	30.8
10 Pooled Applied to all data	392	391	388	5372	
11 Difference between intercepts			2	16	8.0

Comparison of slopes: $F = 30.8/13.6 = 2.26$ (d.f. = 6,380)
Difference not significant at $2\frac{1}{2}\%$ level

Comparison of Intercepts: $F = 8.0/13.9 = 0.58$ (d.f. = 2,386)
Difference not significant at 5% level

7.10 Conclusions

This chapter has described a detailed regression analysis of the I.C. Kronos system. In particular, the short job workload was modelled extensively. As a result of this work, the purely regression Workload Model of the system was constructed.

The initial models of the short job workload, in the presence of a timesharing load, were unsatisfactory. An analysis of residuals revealed certain discrepancies, namely large positive residuals for jobs with large elapsed times, at certain periods of the day. A close study of the Dayfile revealed that during these periods, the system was heavily loaded. When these large residuals were excluded, models with much better fits were constructed.

A further analysis was carried out while the system was supporting an entirely batch workload. In the absence of the timesharing load, much better fits were obtained. However, to build models with consistent regression coefficients, some unrepresentative jobs (less than 5% of the total sample) had to be excluded.

Four models were constructed, representing four different sessions, and compared. The results showed that the regression coefficients of the four models were consistent, although there was a greater variation in the intercepts. Hence it is legitimate to pool the data for all four sessions to construct the regression Workload Model:

$$Y = 3.25 + 2.25CPU + 1.08NCC + 5.32AVJ$$

The Workload Model is a satisfactory representation of the four sessions, which span a period of four months. This is considered a significant result. Hitherto, both in this project and elsewhere, the inconsistency of regression coefficients has been a major problem experienced in apply-

ing regression modelling techniques to computer system performance evaluation. Bard (B1, B2) and Schatzoff/Bryant (S2) have described how they were unable to build models with consistent regression coefficients.

The Regression Coefficients

As the regression coefficients of the models have been shown to be consistent, an attempt may now be made to interpret their values. The regression coefficient of the CPU variable is an estimate of the average time expansion factor experienced by each job for each second of CPU time. This time may be partly due to system overhead and partly due to I/O activity, as no I/O term appears in the model. The coefficient of NCC is an estimate of the overhead to initiate a job step, which is a significant overhead for short jobs. The coefficient of AVJ is an estimate of the average delay experienced by a short job due to competition from other short jobs. The intercept is an estimate of the fixed overheads associated with a job, such as job initiation and job termination time.

The regression analysis of very lightly loaded situations (7.9) and very heavily loaded situations (7.6) has shown that the regression coefficients vary in these extreme situations. The regression coefficients of the CPU and NCC variables are lower in lightly loaded cases, representing lower system overheads, and higher in heavily loaded cases, representing higher system overheads.

Limitations of the Model

The standard error of the residuals, 7.8, is high compared with the mean of the dependent variable, short job elapsed time, which is around 20 seconds. The main reasons for the high standard error are:

(a) The measure of the load on the system is only approximate. This is reflected in the greater fluctuation of the regression coefficient of the average short job load variable, AVJ.

(b) No measures of job rollout time are available. Hence when rollin/rollout activity is high, for example when the timesharing load is heavy, the predictions of the model are poor.

(c) No good measures of I/O demand are available. Hence, the predictions of the model are poor for those jobs whose I/O demands differ greatly from the average.

These limitations are due to the limitations of the data used in constructing the models. A further limitation of the Workload Model is a structural one. It is necessary for one of the independent variables to reflect system load, so that varying loads may be modelled. However, as regression models are static (3.4), estimates of the load on the system must be specified in advance of a run of the model.

One method of overcoming this limitation is to develop a hybrid model in which simulation techniques are combined with regression techniques. By this means, the model is able to adjust its estimate of system load as each modelled job executes. This approach is described in the next chapter.

CHAPTER 8: THE LOAD ADJUSTING MODEL

8.1 Introduction

This chapter considers the limitations of the purely regression Workload Model and attempts to overcome some of them by developing a more detailed hybrid model, the Load Adjusting Model, in which simulation techniques are introduced and combined with the regression techniques. The Load Adjusting Model has been applied to the modelling of the Imperial College Kronos system at the second level of detail.

Section 8.2 discusses the limitations of the Workload Model and describes different methods of developing fast dynamic models of computer systems. Section 8.3 describes the concepts of the Load Adjusting Model (LAM). In section 8.4, the application of the LAM to the Imperial College Kronos system is considered. The design and implementation of the model are described in sections 8.5 and 8.6 respectively. The calibration of the model is described in sections 8.7 and 8.8; the methodology in 8.7 and the results in 8.8. Finally the validation of the model is described in section 8.9.

8.2 Fast Approximate Models of Computer System Performance

8.2.1 Limitations of the Workload Model

Once the purely regression Workload Model has been calibrated and validated, it may be used for making fast and approximate predictions of a batch job's elapsed time, given the job's resource demands and the load on the system during its execution.

The Workload Model suffers from an important structural limitation. This is because a regression model is static and hence does not recognise the passage of time. To enable the Workload Model to model varying loads, it is

necessary for one of the independent variables to be a measure of the load experienced by a job during execution.

Because the Workload Model is static, the estimate of the load on the system for a given job must be input at the start of a run of the model and cannot be adjusted during the run (figure 8.1). However, in an experimental run, in which the environment of the model (e.g. workload, system parameter settings, etc.) is different from that used during calibration, an accurate estimate of the load experienced by each job is not possible in advance of the run. Consequently, further errors will be introduced into the model.

A model which overcomes this limitation is one which dynamically adjusts its estimates of the load on the system as each modelled job commences or terminates execution (figure 8.2). Such a model must be capable of modelling the passage of time, which a regression model does not.

8.2.2 Fast Dynamic Computer System Models

There are a number of ways of building fast dynamic models of computer systems. One approach is to use analytical models of which the most widely used are queuing models. As pointed out in 3.3.2, however, queuing models usually involve a number of simplifying assumptions to make them more amenable to mathematical analysis. The main assumption often employed is that the request inter-arrival time distribution follows an exponential distribution, which assigns the highest probability density to the smallest time interval of length zero. This assumption is often suspect because of the finite source nature of the arrival process (B14).

In the Imperial College system, short jobs are submitted by users, mainly via the local cafeteria service and also via remote job entry stations. The mean short job elapsed time is around 20 seconds (table 7.16), while the

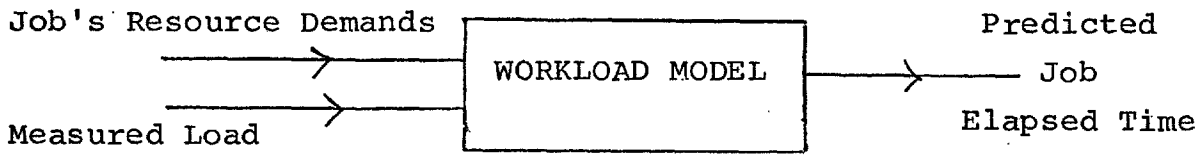


Figure 8.1: The Workload Model

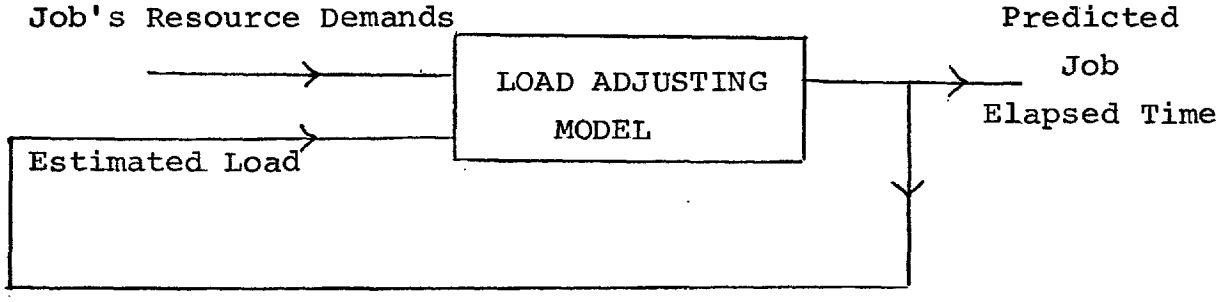


Figure 8.2: The Load Adjusting Model

time taken to read a 200 card deck is 10 seconds on the local card reader, assuming perfect operation, and much longer via remote card readers. Consequently short values of inter-arrival time are very unlikely, and the assumption that the request inter-arrival time is exponentially distributed is not valid.

An alternative method is to use simulation models. However, a simulation model which produced results similar to the Workload Model would probably need to model the system in considerably more detail, and consequently be more expensive to implement. A more promising alternative is to combine simulation with different modelling techniques to produce dynamic hybrid models.

Kimbleton has described an analytically driven computer system simulator (K3) which combines simulation and queuing modelling techniques. The model is trace-driven. A modelled system session consists of a series of time segments, where a time segment is terminated by a job arrival or termination event. Hence the number of processes (jobs) executing in a time segment is constant. All the processes are assumed statistically identical. Time segment statistics are predicted analytically and aggregated using simulation techniques. The model is fast and is said to have compared well in a very limited test with a trace-driven simulation model of the same system. The test consisted of both models processing five identical jobs which started simultaneously. Kimbleton states that as the jobs become progressively less statistically identical, the results become poorer.

The method proposed here for developing dynamic hybrid models of computer systems is to combine regression techniques which are static with simulation techniques which are dynamic. One essential requirement of this method is the construction of a simulation framework which models the passage of time.

By this means, the static regression model is converted into a dynamic hybrid simulation/regression model (see also 3.5.4).

8.3 Concepts of the Load Adjusting Model

8.3.1 Introduction

This section describes the concepts of the hybrid Load Adjusting Model (LAM) which combines simulation and regression modelling techniques to predict job elapsed time, under varying load conditions. The Load Adjusting model is trace driven and models the execution phase of a batch job, that is the time from when a job is first scheduled for execution to the time it terminates. The time spent in this phase is the job elapsed time.

In the Workload Model, the load experienced by each job is input as an independent variable, and is used to compute the delay experienced by the job due to competition from other jobs. In the Load Adjusting Model, a simulation framework is created which allows each job's progress through the system to be modelled dynamically.

A regression submodel predicts each job's elapsed time in the absence of competition from other jobs. The simulation framework allows the number of jobs in execution at any stage to be estimated. A numerical submodel estimates the time delay experienced by a job due to the competition from other jobs, for each period when the number of jobs executing is constant. The simulation framework maintains a running sum of the predictions of the two submodels. At the simulated time of job termination, this sum is the predicted job elapsed time.

8.3.2 Modelling Job Elapsed Time

A batch job's elapsed time t_e may be considered as consisting of two terms:

$$t_e = t_j + t_d \qquad (1)$$

t_j is the elapsed time a job would experience if no other job were competing with it for resources. t_d is the delay a job experiences due to competing with other jobs for system resources. t_d is equal to zero if the job experiences no competition from other jobs. Consequently t_j is the minimum elapsed time a job would experience in the system. t_j will be referred to as the job execution time from now on.

In a purely regression model, both t_j and t_d are predicted using regression techniques. In the hybrid model, t_j is also predicted using regression techniques. However t_d is predicted dynamically using the simulation framework.

The job execution time t_j is a function of a job's resource demands and may be predicted by the regression submodel:

$$t_j = f(d_1, d_2, \ldots d_n) \qquad (2)$$

where $(d_1, d_2 \ldots d_n)$ are the job's resource demands, e.g. CPU time, memory and I/O demands.

8.3.3 Time Segments

The Load Adjusting Model is a dynamic model and therefore explicitly recognizes the passage of time. The LAM is capable of modelling a whole system session. The modelled session is divided up into a series of time segments.

A time segment is defined as an interval of time during which the number of jobs competing for resources is constant. A time segment is started or terminated by one of two possible events:

a) arrival of a job
b) termination of a job.

To simplify the model, it is assumed that in any time segment t_i, in which there is more than one job competing for resources, each job is treated identically by the system.

In segment t_i, each executing job experiences some useful execution t_{ji} and some delay t_{di}. t_{ji} may be accounted for by CPU time, I/O time, or by the system carrying out some function for the job, e.g. job step initiation. t_{di} is the delay experienced by a job due to the competition from other jobs for scarce system resources, and so may represent time waiting for CPU, waiting for I/O, or time rolled out of Central Memory.

A job's elapsed time t_e may be expressed as:

$$t_e = t_j + t_d$$

$$= t_j + \sum_{i=1}^{s} t_{di}$$

where s is the number of time segments a job goes through in the execution phase.

For each job, t_j may be predicted at the simulated time of job arrival using equation (2). t_{di} is estimated for each time segment as described next.

8.3.4 Modelling Delay Time

It is assumed that in each time segment, all jobs are treated identically by the system. It is further assumed that the delay t_{di} experienced by each job in a time segment is:

(i) a function of the number of jobs, n, competing for resources with a given job

(ii) a linear function of the length of the time segment t_i

i.e. $t_{di} = t_i \, g(n)$ (3)

In the general case, we assume that g(n) is a polynomial of the form:

$$g(n) = a_o + \sum_{k=1}^{p} a_k \, n^k$$

However, since t_{di} has been defined such that there is no delay if only one job is executing,

i.e. $t_{di} = 0$ when $n = 0$

$\therefore a_o = 0$

and $g(n) = \sum_{k=1}^{p} a_k \, n^k$

Furthermore, if we assume that only the first two terms of the polynomial are significant, then we have for any time segment t_i in which there are N jobs executing.

$$g(N) = a_1N+a_2N^2$$

Substituting for g(N) in 3:

$$t_{di} = (a_1N+a_2N^2)t_i \qquad (4)$$

However $t_{ji} = t_i-t_{di}$

Substituting for t_{di} from (4):

$$t_{ji} = (1-a_1N-a_2N^2)t_i \qquad (5)$$

$$t_i = \frac{t_{ji}}{1-a_1N-a_2N^2} \qquad (6)$$

8.3.5 Estimating Time Segment Length

A time segment is terminated either by:

a) a new job arriving
b) a job terminating

The time of the next job arrival t_a is obtained from a trace. The time a job terminates is estimated by the model.

When a job enters the system, its execution time t_j is predicted by the regression submodel (equation 2). At the start of each time segment, each job in the system has a remaining execution time t_{jr}, which is the real time a job would require to complete execution if no other jobs

were competing for resources. If we assume all jobs are treated identically, then the job with the minimum t_{jr} (given by t_{jrm}) is the job that will terminate first. The time segment t_{im} necessary to complete execution of the job with execution time t_{jrm} is computed using equation (6).

$$t_{im} = \frac{t_{jrm}}{1-a_1N-a_2N^2} \qquad (7)$$

t_{im} is then compared with t_a to determine whether the next event is a job arrival or a job termination. Hence, the length of the next time segment t_i is given by

$$t_i = \min(t_{im}, t_a)$$

Given t_i, the execution time t_{ji} and delay time t_{di} for this segment may be evaluated using equations (5) and (4) respectively. t_i is added to the value of the elapsed time so far (t_{es}) for each job and t_{ji} is subtracted from the value of the execution time remaining (t_{jr}) for each job.

$$t_{es}' = t_{es} + t_i$$

$$t_{jr}' = t_{jr} - t_{ji}$$

This procedure continues until t_{jr} is reduced to zero for a particular job. This represents the time at which the model estimates the job will terminate. The accumulated elapsed time at the simulated time of job termination is then the predicted elapsed time for that job.

8.4 The Load Adjusting Model of the Kronos System

8.4.1 The Regression Submodel

In this section, the application of the Load Adjusting Model to the Imperial College Kronos system is considered. In particular, the short job workload on the system is modelled using this method.

The Workload Model, developed for the short job workload in Chapter 7, is a regression equation of the form:

$$t_e = b_o' + b_1'T + b_2'K + b_3'N \qquad (8)$$

where t_e is the job elapsed time
T is the CPU time required
K is the number of job steps
N is the average number of short jobs in competition with this job over its lifetime.

It is shown in Chapter 7 that the independent variables of equation (8) are not correlated. Hence, when N = 0, a suitable form for the equation is:

$$t_e = b_o' + b_1'T + b_2'K$$

Furthermore, when a subset of the short job workload was modelled, namely those jobs which did not experience any competition from other jobs, the fitted equation was:

$$t_e = b_o'' + b_1''T + b_2''K \qquad (9)$$

Since only the short job workload is modelled in this case, it is appropriate to amend the definition of job

execution time given in 8.3.2. The job execution time is now defined as the elapsed time experienced by a job when it experiences no competition from other short jobs. It is clear that an appropriate model for short job execution time is then:

$$t_j = b_o + b_1 T + b_2 K \qquad (10)$$

Consequently, this is the regression submodel used in the Load Adjusting Model to predict the job execution time. The choice of the most appropriate values of the coefficients b_o, b_1 and b_2 is left to the calibration process, which is described in 8.7.

8.4.2 Assumptions made by LAM

A number of simplifying assumptions are made in applying the Load Adjusting Model to the Kronos system. These are:

1) The LAM has been applied to a subset of the workload on the system, namely the short job workload. The delay experienced by a short job in a time interval is assumed to depend only on the competition from other short jobs. The competition from long jobs is ignored. This is clearly a limitation of the model, but the assumption is similar to that made by the Workload Model.

2) It is assumed that in each time segment, where the number of jobs executing is constant, all jobs are treated identically by the system. This is a reasonable assumption for CPU allocation, where a round-robin scheduling algorithm is enforced. It is likely to be less reasonable for I/O management.

3) Since all real time data obtained from the Dayfile (e.g. job start and end times) are measured in units of a second, the basic real time quantum in the model is the second. It should be pointed out, however, that CPU time is measured in milliseconds. This value is used by the regression submodel to estimate job execution time, which is then rounded to the nearest second.

These assumptions are bound to lead to inaccuracies in the model. Attempts are made to minimize these during the calibration of the model.

8.5 Design of the Load Adjusting Model

8.5.1 Overview

The central part of the Load Adjusting Model (LAM) is a co-ordinating routine. The co-ordinating routine has a loop which it goes round once for every time segment. At the start of the loop, the simulation clock is set to the time of the event processed in the previous round, which constitutes the start time of the current segment. The first task of the co-ordinator is to determine what the next event is and the time at which it occurs. This is carried out as described in 8.3.5, that is the time of next job arrival is compared with the time of the first job to terminate. Provision must also be made for detecting an end-of-session event.

Once the time of the next event has been determined, the simulation clock is advanced to this time, which represents the end of the time segment. The statistics for all executing jobs may now be updated for this time segment. The event occurring at this simulated time is then processed. It is possible for more than one event to occur at the same simulated time, e.g. job termination and job arrival, and

LAM deals with this accordingly. Once the event(s) have been processed, this constitutes the end of this round of the main loop, and the process is repeated for the next time segment.

8.5.2 Description of LAM

A block diagram showing the overall design of the model is displayed in figure 8.3. A short description of each of the routines follows:

MAIN — is the main routine. It carries out all necessary initialisation and then enters COORD.

COORD — is the co-ordinating routine for the model. It goes round a loop once for each time segment. First it calls SENEVT to determine what the next event is, and to determine the length of the next time segment. It then calls UPDENT to update the entries of all jobs currently in execution. Finally, it calls one (or more) of the three event processing routines.

SENEVT — is called by COORD to determine what the next event is, and the estimated time at which it occurs. The next event could be a job arrival, job termination or end of session. It calls SRINT to estimate what the minimum time is for the next job to terminate. This is compared with the next job arrival time, to decide what the next event is. However, if the number of jobs currently in execution is equal to a certain limit, then no jobs will be allowed to commence execution, until the number of jobs in execution falls below this limit.

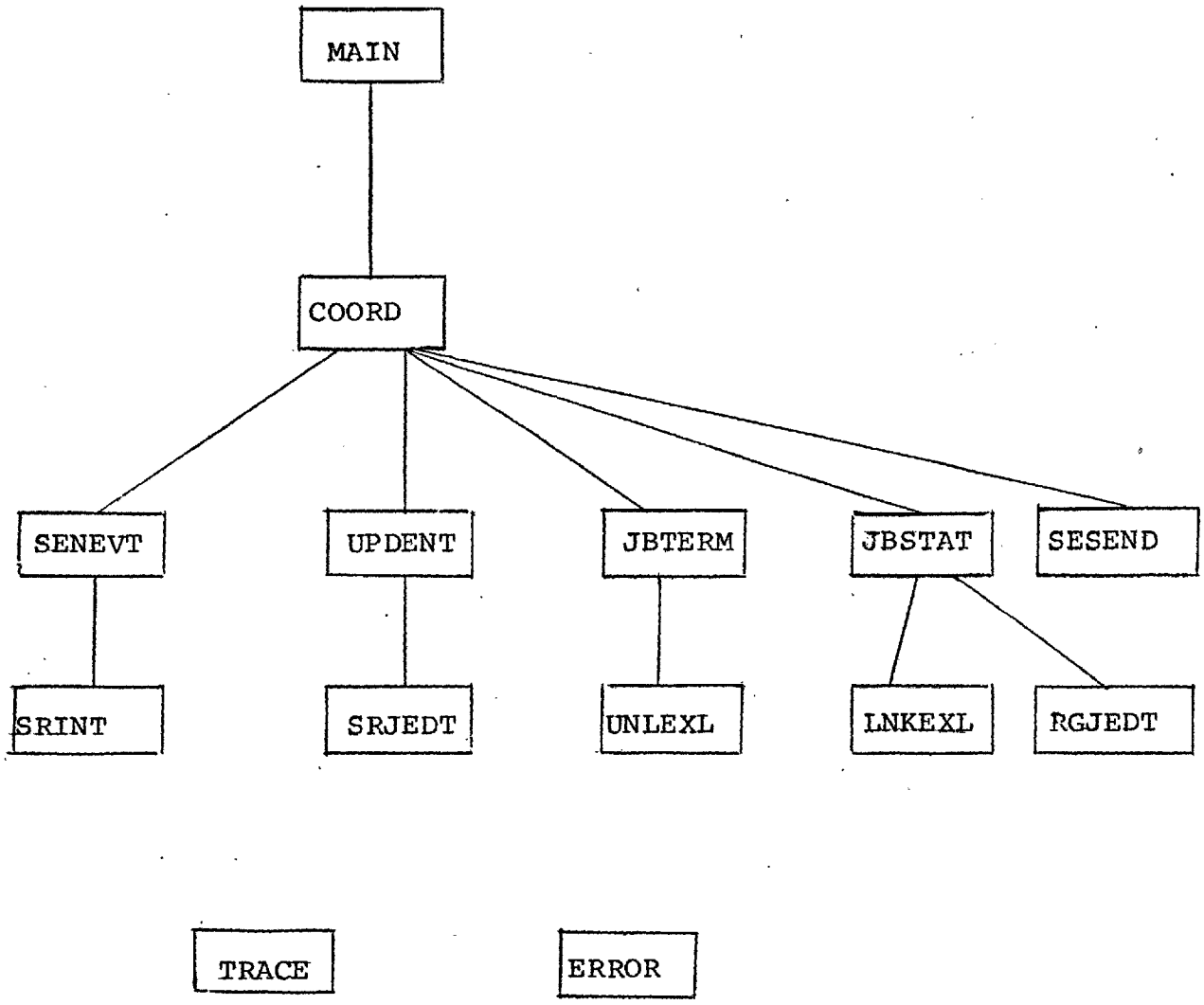


Figure 8.3: Structure of Load Adjusting Model

UPDENT Given the length of the next time segment t_i, UPDENT calls SRJEDT to compute the delay time t_{di} and the execution time t_{ji} for the segment. A linked list is maintained which contains each job in execution state. For each job on the list :

(a) the delay time for this interval is added to the cumulative delay so far.
(b) the execution time for this interval is decremented from the remaining execution time.

JBTERM handles job termination. The entry for the job terminating is taken off the Execution list. The accumulated job statistics are copied into a job output buffer. If the buffer is full, it is output to a disc file.

JBSTAT handles job initiation. An entry is set up for the new job and linked onto the Execution list. The job's resource requirements and other information are read from the job's entry in the input buffer and copied to the Execution list entry. RGJEDT is called to predict the job's execution time. If the input buffer is now empty, it is replenished.

SESEND is called at the estimated time of the end of session. It outputs the final bufferful of data to disc, and outputs statistics of the run to the line printer.

UNLEXL(N) searches the execution list for job N. When the entry for job N has been found, it is taken off the list. UNLEXL then clears the contents of the entry and links it onto a free list.

LNKEXL(N) removes an entry off the free list and links it onto the end of the Execution list.

RGJEDT uses equation (10) to compute the predicted job execution time t_j for a job commencing execution.

SRINT uses equation (6) to compute the estimated real time required to complete execution of a given job for a given load on the system.

SRJEDT computes the estimated job execution time t_{ji} (using equation 5) and the estimated delay time t_{di} (using equation 4) for a given time segment t_i and load N on the system.

TRACE outputs a trace message each time it is called, providing the trace flag is on. The message is either for a job commencement or job termination.

ERROR (I) is the error routine which may be called from a number of places in the model. It outputs an error message, dumps the contents of various locations and arrays, and stops. This routine is especially useful during testing, but is also capable of detecting any errors in the input data.

8.6 Implementation

8.6.1 Introduction

A preprocessor prepares a workload trace for input to the Load Adjusting Model. The predictions of the LAM are analysed by a postprocessor.

Three sessions were used in the calibration and validation of LAM. These were the morning sessions of 27/1/75 and 30/4/75 and the afternoon session of 30/1/75. All three sessions were used in the construction of the Workload Model, described in sections 7.7 to 7.9 of Chapter 7.

8.6.2 Preprocessing

A block diagram showing the steps involved in running the model is shown in figure 8.4. The Dayfile processing programs take as input the Kronos Account and System Dayfiles and output two files, the B and J files (chapter 5). The B file, which is ordered by job termination, consists of a job summary record for each job processed in the session. Each summary record holds measures of the resources demanded by the job during its execution. The J file, ordered by job commencement, contains various measures of the loading on the system during each job's lifetime.

The B and J files are input to the Preprocessor which merges and sorts the two files and outputs the G file, in which job summary records are ordered by job arrival. Only short jobs are selected for the G file. The G file is the input file to the Load Adjusting Model. It is input to the model in the form of a workload trace. Each job is represented by a vector of its resource demands which is input to the model at the simulated time of job arrival.

8.6.3 The Load Adjusting Model

The Load Adjusting Model is coded in Fortran and is well under 500 statements in length. It processes a four hour session on the CYBER, consisting of about four hundred short jobs and about 800 time segments, in under 10 seconds CPU time on the same system.

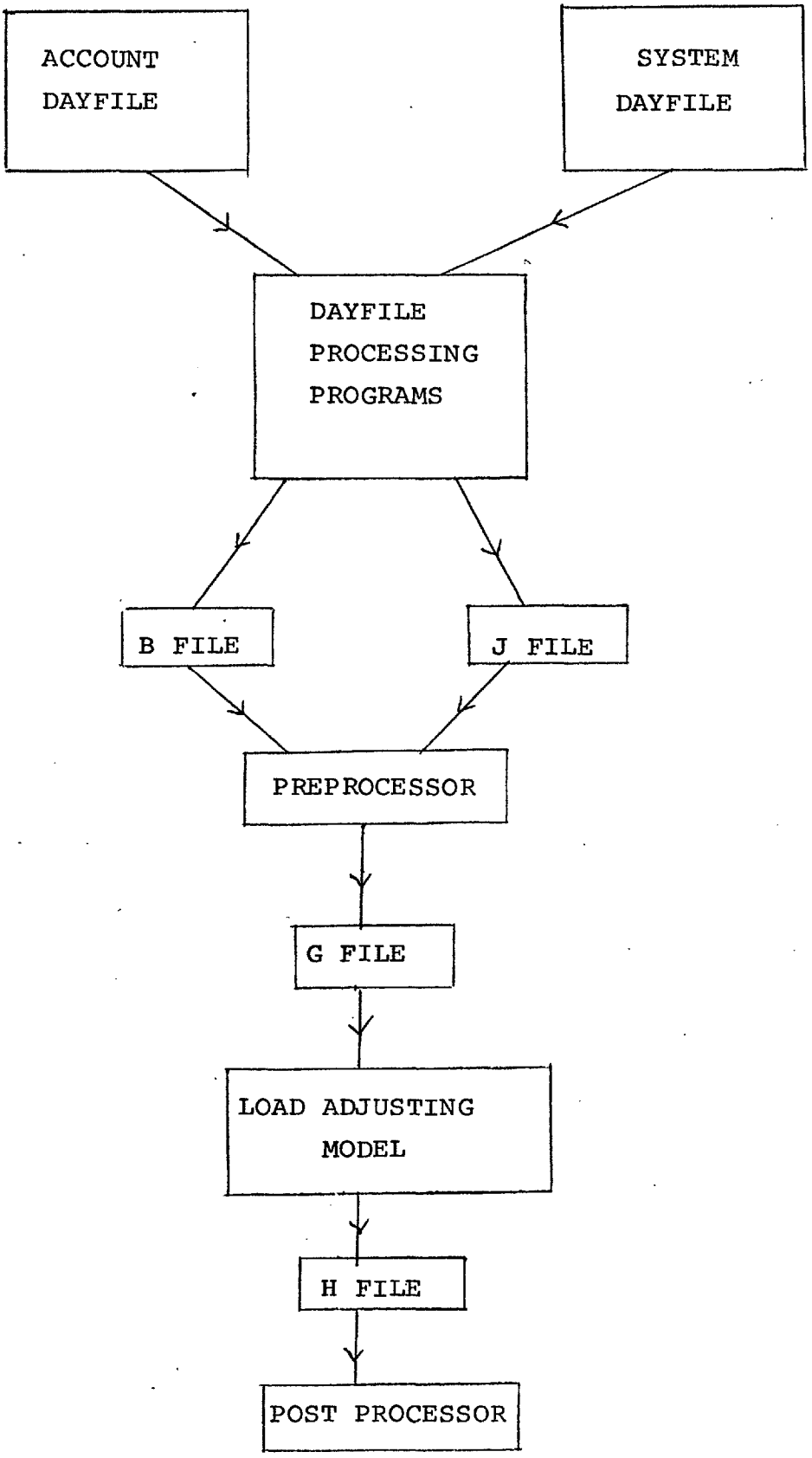


Figure 8.4: Overall Diagram of Dayfile Processing and System Modelling

For a particular run of the model, various parameter settings are input as data to the model. A workload trace is also input to the model in the form of a set of short jobs ordered by job arrival time. The model outputs the following predictions for each job in the form of a disc file called the H file:

a) predicted job elapsed time.
b) predicted job execution time (i.e. estimated elapsed time a job would have experienced had there been no competition from other jobs).
c) predicted delay time a job experiences due to competition from other jobs.
d) predicted average load a job experiences during its modelled lifetime in the execution stage.

The elapsed time predicted by the Workload Model for the same job is also output.

8.6.4 The Postprocessor

The postprocessor analyzes the H file generated by a run of LAM. The postprocessor prints the results in tabular form, computes the means and standard deviations of the predictions of the model, and plots various figures as required. It also carries out a statistical analysis of the results.

The residual for each job, that is the difference between the actual elapsed time and the elapsed time predicted by LAM, is computed by the postprocessor. The residuals are plotted against the following variables:

a) actual job elapsed time.
b) predicted job elapsed time (an example is shown in figure 8.7).
c) estimated average load experienced by the job.

d) predicted delay time for the job (an example is shown in figure 8.6)

e) CPU time used by job (examples are shown in figures 8.5 and 8.8).

f) number of job steps.

8.7 The Calibration Methodology

8.7.1 Introduction

Calibration is an iterative procedure whose objective is to reduce the difference in behaviour between the model and the real system by adjusting the parameters of the model (B4).

There are two sets of parameters which may be adjusted during calibration:

a) The parameters b_o, b_1 and b_2 of the regression submodel for job execution time

$$t_j = b_o + b_1 T + b_2 N \qquad (10)$$

b) The parameters a_1 and a_2 of the submodel for the delay time each job experiences in a time segment:

$$t_{di} = (a_1 N + a_2 N^2) t_i \qquad (4)$$

Since $t_{di} < t_i$, the following restrictions apply:

$$0 \leq a_1 < 1$$
$$0 \leq a_2 < 1$$
$$a_1 + a_2 < 1$$

The overall calibration approach was based on that used in the calibration of a simulation model of OS/360 under LASP(B7).

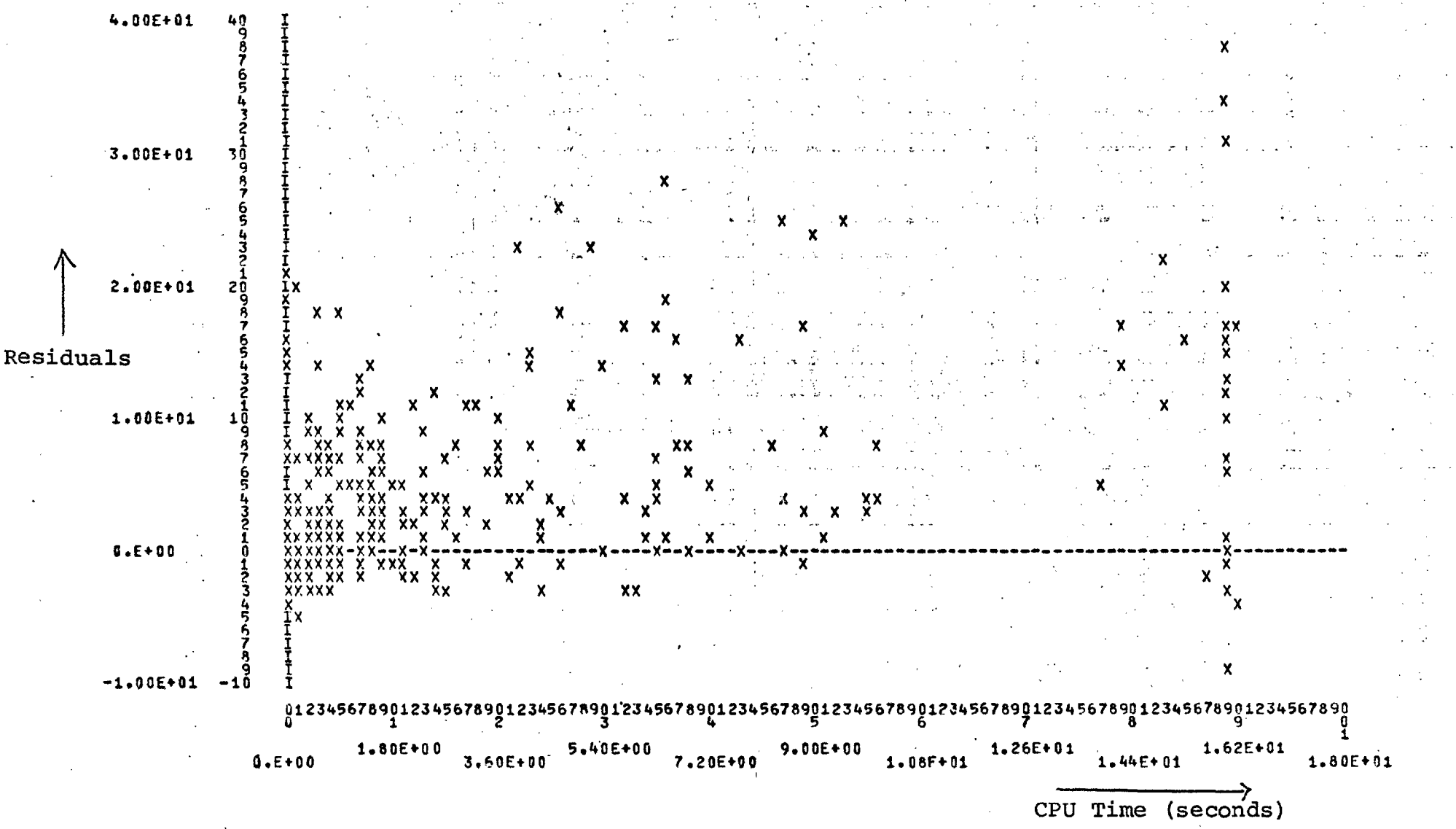


Figure 8.5: Plot of Residuals Against CPU Time (30/4/75 Run 1)

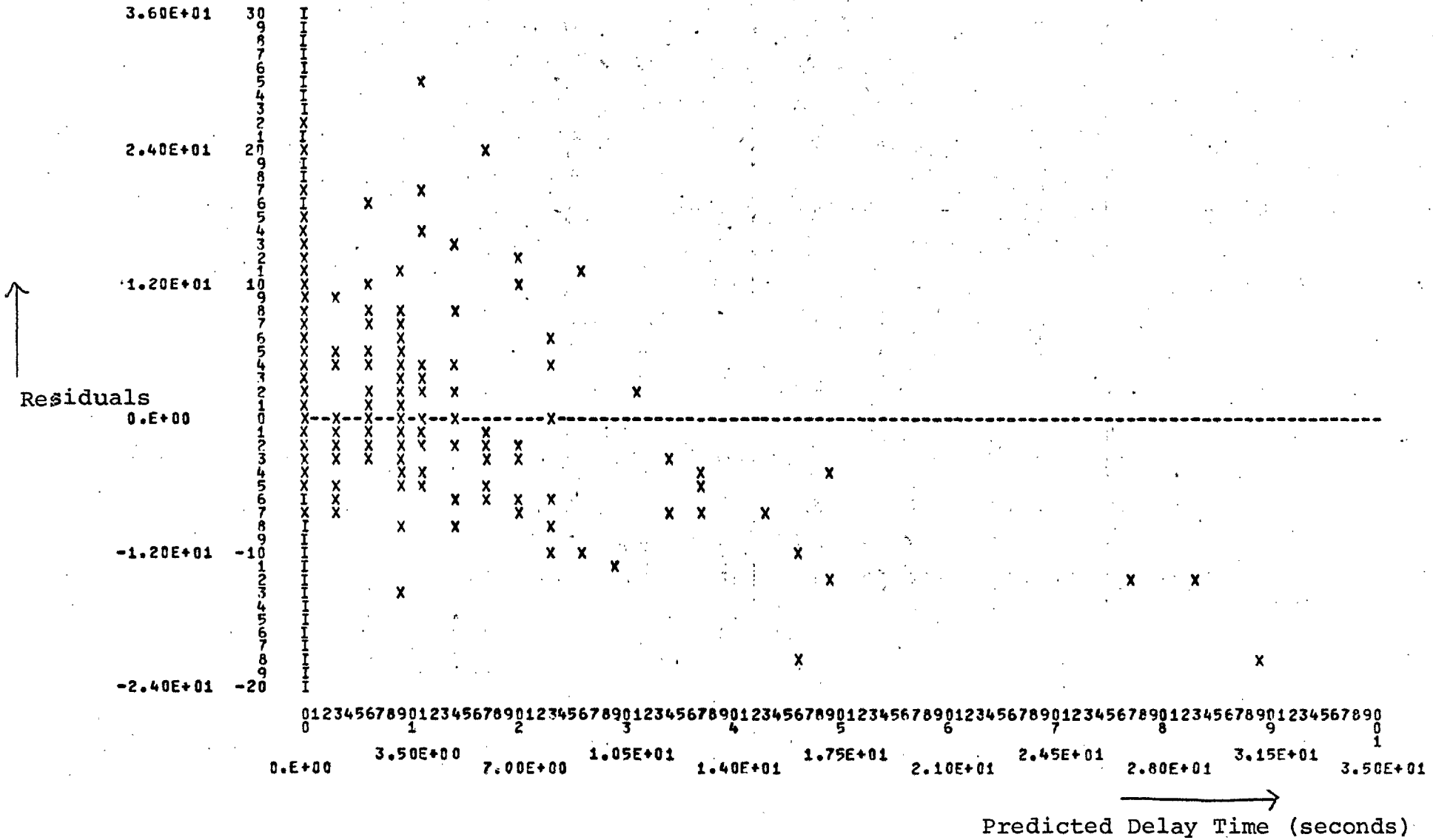


Figure 8.6: Plot of Residuals against Predicted Delay Time (30/4/75 Run 2)

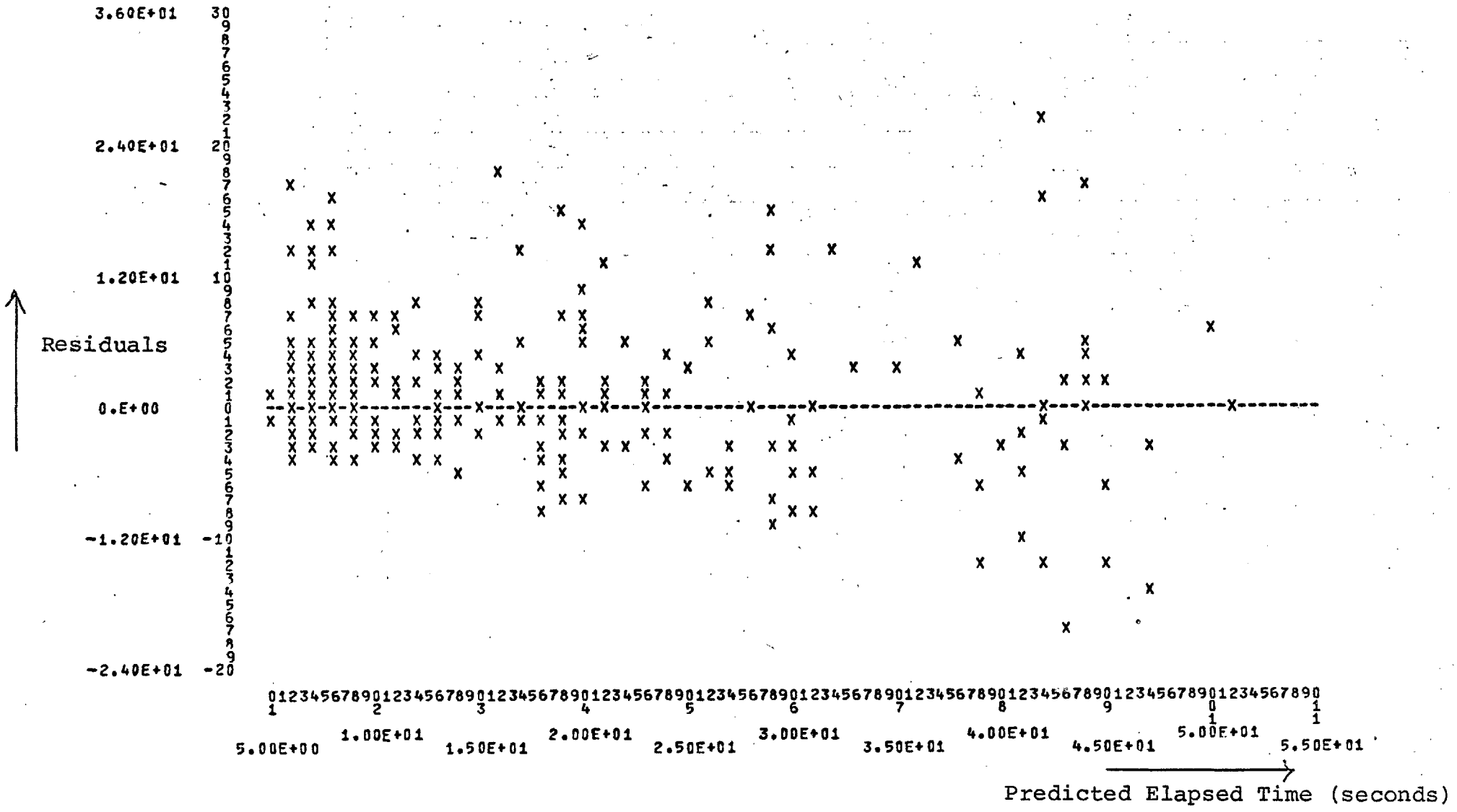


Figure 8.7: Plot of Residuals against Predicted Elapsed Time (30/4/75 Run 6)

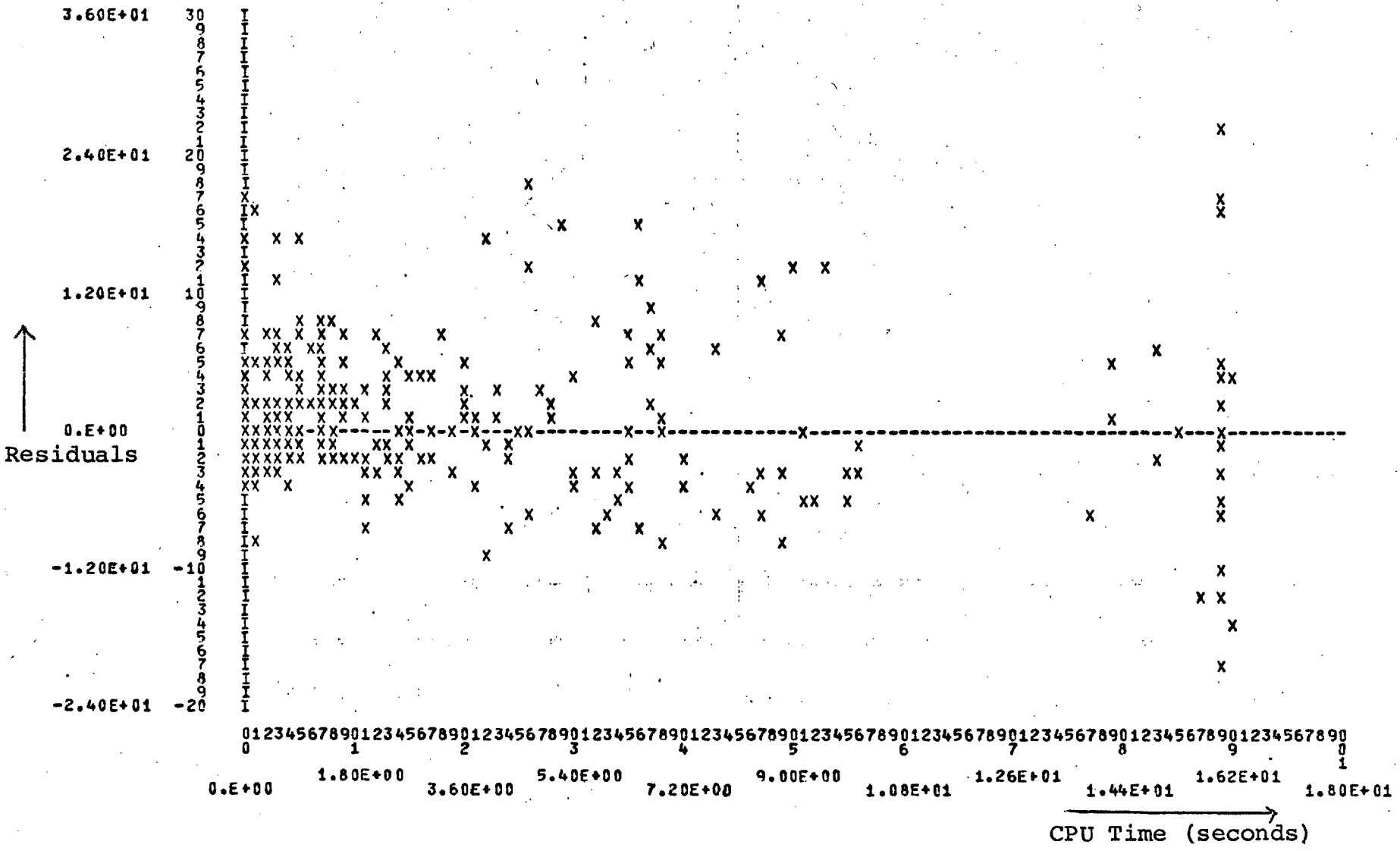


Figure 8.8: Plot of Residuals against CPU Time (30/4/75 Run 6)

8.7.2 Figures of Merit

Since the objective of the calibration exercise is to reduce the difference in behaviour between the model and the real system, some 'figure of merit' is necessary for deciding whether one model or version of a model is 'better', i.e. significantly closer in its predictions to the real world, than another.

One method of comparing the difference between the real system and the model is to compare the real job elapsed time with the predicted job elapsed time. The difference between these values is the residual. However, this measure has the disadvantage, when averaged over all jobs, that residuals of the type 'job a is x seconds too fast while job b is x seconds too slow' cancel out. Consequently, more satisfactory measures are those which do not consider the sign of the residual, such as the absolute value of the residual or the residual squared.

Developing the Workload Model has shown that large residuals occur because of situations which are not taken into account by the model, due to the limitations of the available data. Using the residuals squared as a figure of merit places a larger emphasis on these large residuals. Since basically the same data is used for developing LAM, it was decided to use the absolute value of residuals as the figure of merit.

8.7.3 A Significance Test

For each run of the model in calibration, it is likely that the predicted elapsed time will be better for some jobs and worse for others, as measured by the figure of merit. Each time a reduction in the mean absolute value of residuals is achieved, it is necessary to determine whether the reduction is a significant one. For this reason, a statistical significance test is used.

Developing the Workload Model has shown that the distribution of residuals is usually non-normal. Consequently a non-parametric significance test was used in the calibration of LAM, i.e. one that does not assume that the residuals are normally distributed. The test is the Wilcoxon matched-pairs signed-ranks test (S8).

Early attempts at calibrating LAM using the Wilcoxon test showed that there was always a very significant difference between the model predictions and the real system behaviour. Consequently, it was decided to use the Workload Model as a standard for comparing LAM predictions. For each test, LAM and WM are run using the same input trace for a given session. The elapsed time for each job in the session is predicted by both models and the residuals are derived:

r_s = actual (i.e. measured) job elapsed time - elapsed time predicted by Load Adjusting Model

r_r = actual job elapsed time - elapsed time predicted by Workload Model.

The Postprocessor prepares a sequence of matched pairs, one for each job, of absolute values of residuals ($|r_s|, |r_r|$). The sequence of matched pairs are then compared using the Wilcoxon test, which tests the null hypothesis that there is no significant difference between the models (figure 8.9). It outputs the probability of the hypothesis being true. If the probability is above 10%, then the hypothesis is accepted. If the probability is below 10% (5%), then the hypothesis is rejected at the 10% (5%) significance level, i.e. with 90% (95%) confidence.

8.7.4 The Method of Good Balance

Each time a run with the model is made, a decision has to be made about the settings of the parameters for the next run. Some indication of the sources of inaccuracy of

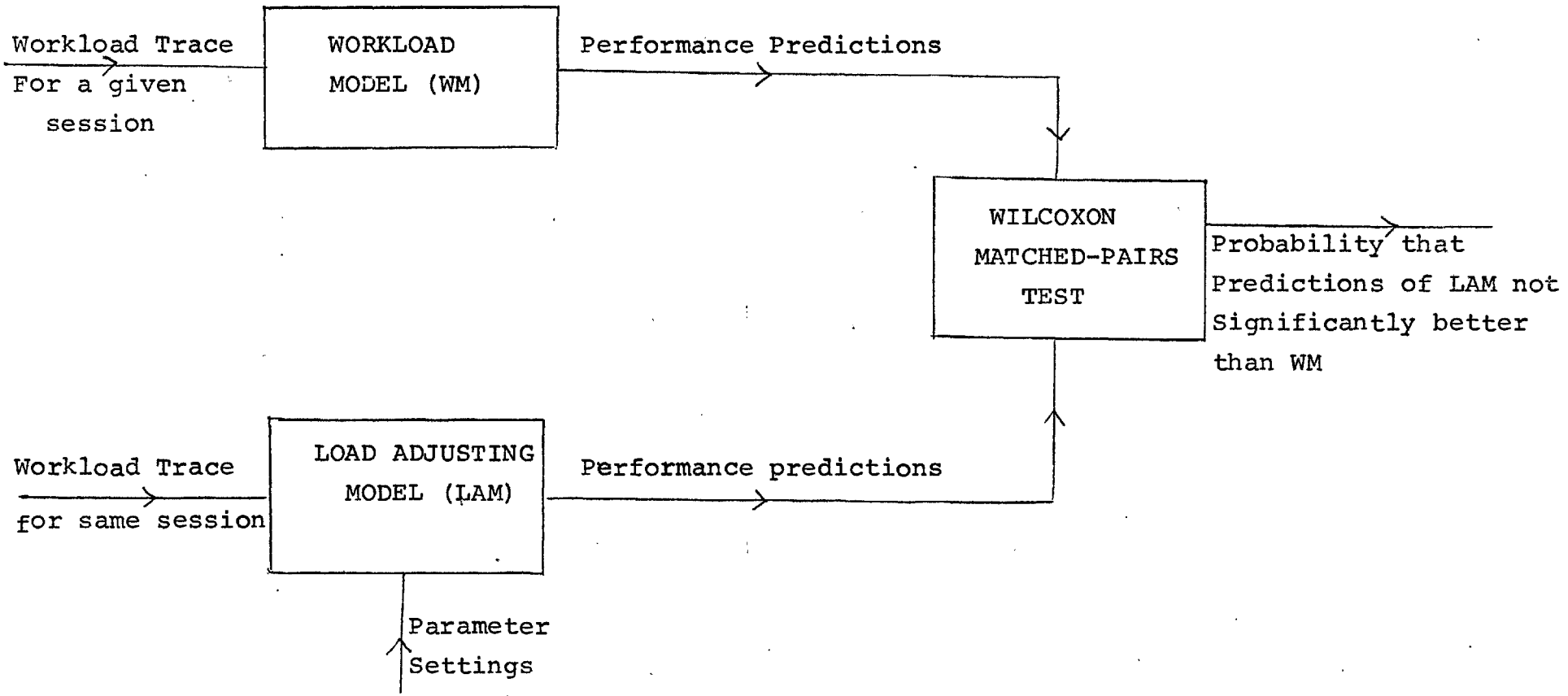


Figure 8.9: Procedure for Calibrating Load Adjusting Model

the model may be obtained by studying the residual plots. A more systematic method is used by the Method of Good Balance which was developed by Beilner (B5).

The Method of Good Balance provides a systematic means of guiding the model-builder towards constructing a well-balanced model. The error in such a model does not depend significantly on any job characteristics or on the system load, but instead is randomly distributed amongst all types of jobs and loads. A linear regression analysis is carried out with the residual in job elapsed time as the dependent variable and measures of job characteristics and system load as the independent variables. The objective of the exercise is to gradually develop a regression equation in which none of the regression coefficients are significant. Once this is achieved, it means that none of the independent variables make a significant contribution to the model, and that the error in the model does not depend significantly on job characteristics or system load.

8.8 The Calibration in Practice

8.8.1 Introduction

After each run of the Load Adjusting Model with a given setting of the parameters, the following analysis is carried out:

(a) The mean of the absolute value of the residuals $|\bar{r}_s|$ is inspected to determine if a reduction in this figure of merit has resulted from the new parameter settings.

(b) The Method of Good Balance indicates whether the residuals are correlated with any of the job or

load characteristics, e.g. CPU time, number of job steps, delay time.

(c) The residual plots show the nature of the relationship between the residuals and the particular job and load characteristics.

(d) The Wilcoxon test indicates how the LAM compares with the Workload Model.

With the guidance provided by this analysis, the parameters are altered and the process is repeated.

8.8.2 Initial Parameter Settings

When calibrating the model, a decision must be made at an early stage about the initial settings of the calibration parameters.

The choice of the initial parameter settings for the regression submodel for execution time

$$t_j = bo + b_1T + b_2N \qquad (10)$$

was guided by the regression models of the short job workload (Chapter 7). In particular, it was guided by the models of the subset of the workload that did not experience competition from other short jobs (table 7.24). As this subset of the workload is not a representative one (see 7.9.1), it is to be expected that the regression coefficients of the model of this subset would not be appropriate for the regression submodel of execution time.

The appropriate parameters need to be determined by means of the iterative tuning process. It was decided

to set each parameter initially to the minimum value of the regression coefficient for the appropriate independent variable in the three models of the subset (table 7.24). Hence, the initial settings were:

$$bo = 4.7 \qquad b_1 = 1.4 \qquad b_2 = 0.5$$

Intuitively reasonable values were chosen for the initial settings of the parameters of the delay time submodel:

$$t_{di} = (a_1 N + a_2 N^2) t_i \qquad (4)$$

The initial settings were $a_1 = 0.15$ $a_2 = 0$.

8.8.3 Analysis of Residuals

Initially, some observations were excluded from the calibration. These were the same observations that had been excluded from the Workload Model, for the reasons described in 7.7.4.

During the calibration, an analysis of residuals revealed that at certain times for each session, large positive residuals were obtained using the Load Adjusting Model, which were not obtained using the Workload Model. Initially, this was puzzling. However a more detailed analysis revealed that these large positive residuals occurred at times when the delay time predicted by LAM was small, while the delay time predicted by the Workload Model was much larger.

A study of the System Dayfile for the times in question, revealed two causes for this.

1. Occasionally system jobs are run. These jobs carry out functions such as Dayfile dumping, permanent file dumps and reloads. System jobs are allocated a higher CPU priority than other

jobs. Hence the presence of system jobs perturbs the systems and sometimes leads to a buildup of short jobs, which are consequently delayed.

2. Some of the jobs excluded from the Workload Model, (and LAM), because of their unusual characteristics, run for much longer than normal and cause a build-up of short jobs with subsequent delays.

Neither of these two types of job are actually used in the calibration, either because their presence is not recorded (system jobs, see 5.5) or because they are deliberately excluded (uncharacteristic jobs). However the delay caused to other jobs is felt, and much more so by the Load Adjusting Model than the Workload Model.

The reason for this is due to the difference in the basic structure of the two models. The Workload Model is a regression model of the form

$$t_e = bo + b_1T + b_2K + b_3N \qquad \text{(equation 8 in 8.4.1)}$$

To a first approximation, the delay time d_r predicted by the model is given by

$$d_r = b_3N$$

where N is the average short job load experienced by a short job while in execution. N is in fact input to the model as one of the independent variables (figure 8.1). Thus any perturbation to the system which leads to a build-up of short jobs will be reflected in an increased value of N and consequently a larger predicted delay d_r.

On the other hand, the delay d_s predicted by the Load Adjusting Model is generated internally by the model (figure 8.2). If the model is completely unaware of the perturbation that occurred, as in cases 1 and 2 above, then it cannot possibly account for it.

It was decided therefore to exclude from the calibration, those observations which were noticeably affected by the perturbation. The calibration was then recommenced.

8.8.4 An Example of the Calibration

In this subsection, an example of the calibration of the Load Adjusting Model (LAM) for one particular session, the morning of 30/4/75, is given. Starting with the initial parameter settings chosen in 8.8.2, a number of runs were carried out. A selection of the runs (there were considerably more than 7) are displayed in Table 8.1. For each run, the following results are displayed:

(a) the parameter settings for the run

(b) the mean predicted elapsed time $\overline{t_e}$

(c) the mean of the absolute residuals $|\overline{r_s}|$

(d) the result of the Wilcoxon test, i.e. the probability that the null hypothesis, that there is no difference between the LAM and Workload Model (WM), as measured by the distribution of the two sets of absolute residuals, is true

(e) the result of the Method of Good Balance (MGB), in particular which variables have significant regression coefficients, and the sign of the correlation.

In Run 1 (the initial settings), the mean predicted elapsed time of 11.9 is much lower than the mean actual elapsed time, 17.0. $|\overline{r_s}|$ is greater than the mean of the absolute WM residuals $|\overline{r_r}|$. The Wilcoxon test shows that WM is significantly better than LAM. The MGB indi-

Table 8.1: Example of Tuning the Load Adjusting Model - 30/4 Workload Trace

Runs with LAM									
Run	b_0	b_1	b_2	a_1	a_2	predicted $\overline{t_e}$	$\lvert\overline{r_s}\rvert$	P	M.G.B.
1	4.7	1.4	0.5	0.15	0	11.9	5.89	0.001	+CPU +NCC
2	4.7	1.7	0.7	0.30	0	15.4	5.05	0.186	-SDEL +CPU
3	4.7	1.9	0.7	0.23	0	15.3	4.64	0.178	-SDEL
4	4.7	1.9	0.7	0.15	0.01	14.8	4.68	0.253	W.B.
5	4.7	2.0	0.8	0.11	0.01	15.3	4.58	0.118	W.B.
6	4.7	2.1	0.8	0.11	0.01	15.7	4.55	0.064	W.B.
7	4.7	2.1	0.8	0.20	0.01	16.7	4.75	0.370	-SDEL

Mean actual (i.e. measured) t_e = 17.0 $\overline{t_e}$ predicted by WM = 17.9
Mean of absolute value of residuals of WM, $\lvert\overline{r_r}\rvert$ = 4.72

Key:
- $\overline{t_e}$: mean elapsed time in seconds
- $\lvert\overline{r_s}\rvert$: mean of absolute value of residuals of LAM
- P : Probability that there is no difference between LAM and WM
- M.G.B. : Method of Good Balance
- W.B. : Well Balanced model
- CPU : CPU time
- NCC : Number of Control Cards
- SDEL : Predicted delay time

} Variables that residuals may be correlated with.

- +/- : sign of correlation

cates that the coefficients of both the CPU and NCC variables, b_1 and b_2 respectively in equation (10), should be increased in value. The residual plot against CPU time (figure 8.5) also indicates that the residuals are positively correlated with CPU time.

In Run 2, b_1 and b_2, in addition to a_1 are increased in value. As a result, the mean predicted elapsed time is increased to 15.4. $|\overline{r_s}|$ is still greater than $|\overline{r_r}|$, but the Wilcoxon test indicates that the difference between the two models is no longer significant. The MGB indicates that b_1 should be increased further. It also indicates that the residuals are negatively correlated with delay time. This is supported by the residual plot against delay time (figure 8.6).

In Run 3, b_1 is increased and a_1 decreased. This time $|\overline{r_s}|$ is smaller than $|\overline{r_r}|$, but the Wilcoxon test indicates that the difference is not significant. The MGB shows that the residuals are still negatively correlated with delay time, although to a lesser extent than before.

In Run 4, only the delay parameters are altered. a_1 is reduced further while a_2 is set to a non-zero value for the first time, so as to introduce a finer degree of tuning. With these settings, MGB indicates that the model is now well balanced, i.e. the error in the model is no longer correlated with any particular job characteristic or load on the system. However, the Wilcoxon test indicates that, although $|\overline{r_s}|$ is smaller than $|\overline{r_r}|$, there is still no significant difference between the Load Adjusting and Workload Models.

From now on, the iterative tuning procedure becomes more difficult. The objective is to continue improving the model by reducing the value of $|\overline{r_s}|$. However, the MGB can no longer help as the model is now well balanced. The

iterative procedure was lengthy, so only the principal results are highlighted here.

In Run 5, b_1 and b_2 have both been increased in value, while a_1 has been reduced. The Wilcoxon test indicates that the null hypothesis can still not be rejected, but the probability of 0.118 is close to the 10% significance level.

In Run 6, b_1 is increased to 2.1. The Wilcoxon test indicates that the null hypothesis can now be rejected at the 10% level, as the probability of the two sets of absolute residuals being drawn from the same population is 0.064. Figures 8.7 and 8.8 show the residual plots against elapsed time and CPU time respectively for these parameter settings.

Attempts to improve the model further by adjusting each of the parameters or combination of parameters (e.g. Run 7) failed. So the parameter settings for Run 6 were taken to be the 'best' values, since they resulted in the closest agreement between the modelled and real worlds, as measured by the figure of merit of the absolute value of the residuals.

With the parameter settings of Run 6, the mean predicted elapsed time is 15.7 seconds as compared with the mean actual elapsed time of 17.0 seconds. It is possible to select parameter settings which allow the mean predicted elapsed time to be closer to the mean actual elapsed time. In Run 7, the delay parameter a_1 is increased to 0.20. The mean predicted elapsed time is now 16.7, which is much closer to the mean actual elapsed time. However, MGB indicates that the model is no longer well balanced, as the residuals are negatively correlated with delay time. Furthermore, the Wilcoxon test shows that the difference between the LAM and WM is no longer significant, because of the increase in $|\overline{r_s}|$. Similar results are obtained if b_1 is increased instead of a_1. This feature of LAM has been

observed with other simulation models. During the calibration of SOUL (Simulation of OS/360 under LASP), it was noted that for the 'best' parameter settings, the mean predicted elapsed time was significantly lower than the mean actual elapsed time (W2).

8.8.5 Comparison of Modelled Sessions

The model was run with three different workload traces, where each trace represented a different session. In addition to the morning of 30/4/75, the morning session of 27/1/75 and the afternoon session of 30/1/75 were used. The model with the trace for each session was calibrated using the methods described in sections 8.7 and 8.8.4. The sessions are compared in table 8.2. The following points should be noted:

(a) During the last part of the monitored period in the 30/1/75 session, a number of system jobs were run. These perturbed the system in the manner discussed in 8.8.3 thereby causing larger positive residuals. It was decided to exclude this latter period from the calibration.

(b) For all three sessions, the Wilcoxon test indicates that the predictions of the Load Adjusting Model are significantly better, at the 10% significance level, than the Workload Model. For the 27/1 session, the model predictions are significantly better at the 5% level.

(c) There are some differences in the 'best' parameter settings for each session, particularly in b_1, the coefficient of the CPU variable and the delay parameter a_1.

Table 8.2: Comparison of Modelled Sessions

Modelled Session	b_o	b_1	b_2	a_1	a_2	actual $\overline{t_e}$(secs)	predicted $\overline{t_e}$ by LAM	$\lvert\overline{r_r}\rvert$	$\lvert\overline{r_s}\rvert$	P
27/1	4.7	2.3	0.8	0.10	0.01	17.4	16.4	5.08	4.45	0.037
30/4	4.7	2.1	0.8	0.11	0.01	17.0	15.7	4.72	4.55	0.064
30/1	4.7	2.0	0.8	0.11	0.01	16.8	15.4	4.42	3.98	0.069

For key refer to table 8.1

8.9 Validation of the Model

8.9.1 Introduction

Validation of the LAM aims at determining the domain of situations for which the model performs with a given accuracy, for an established calibration (B4, B6).

During calibration, the parameters of the model are adjusted with the objective of reducing the difference in the behaviour between the real and modelled worlds for a particular workload trace. When calibrating the model for a given trace, there is always the danger of overtuning the model, that is adjusting the parameters so well that the predictions of the model for the calibration situation are significantly better than for other situations (B6). Consequently, the objective of the validation process is to find a set of parameter values, determined during calibration for a given workload trace, with which the model predictions are not significantly worse for other traces.

8.9.2 Validation

The LAM parameters are set to values obtained in the calibration of the model with a given trace. The model is then run with the other two traces respectively. Next, a non-parametric test, the Mann-Whitney U-test, is carried out to determine if there is any significant difference in the model predictions. The criterion for comparison is the absolute value of residuals $|\overline{r_s}|$. The null hypothesis, that two independent groups of observations have been drawn from the same population, is tested. In this case, the independent groups are the sets of absolute residuals obtained by running the model, with a given set of parameters, using different input traces (figure 8.10). Since there are three traces, the test is carried out in a pairwise manner, comparing two sets of absolute residuals at a time, making three tests in all.

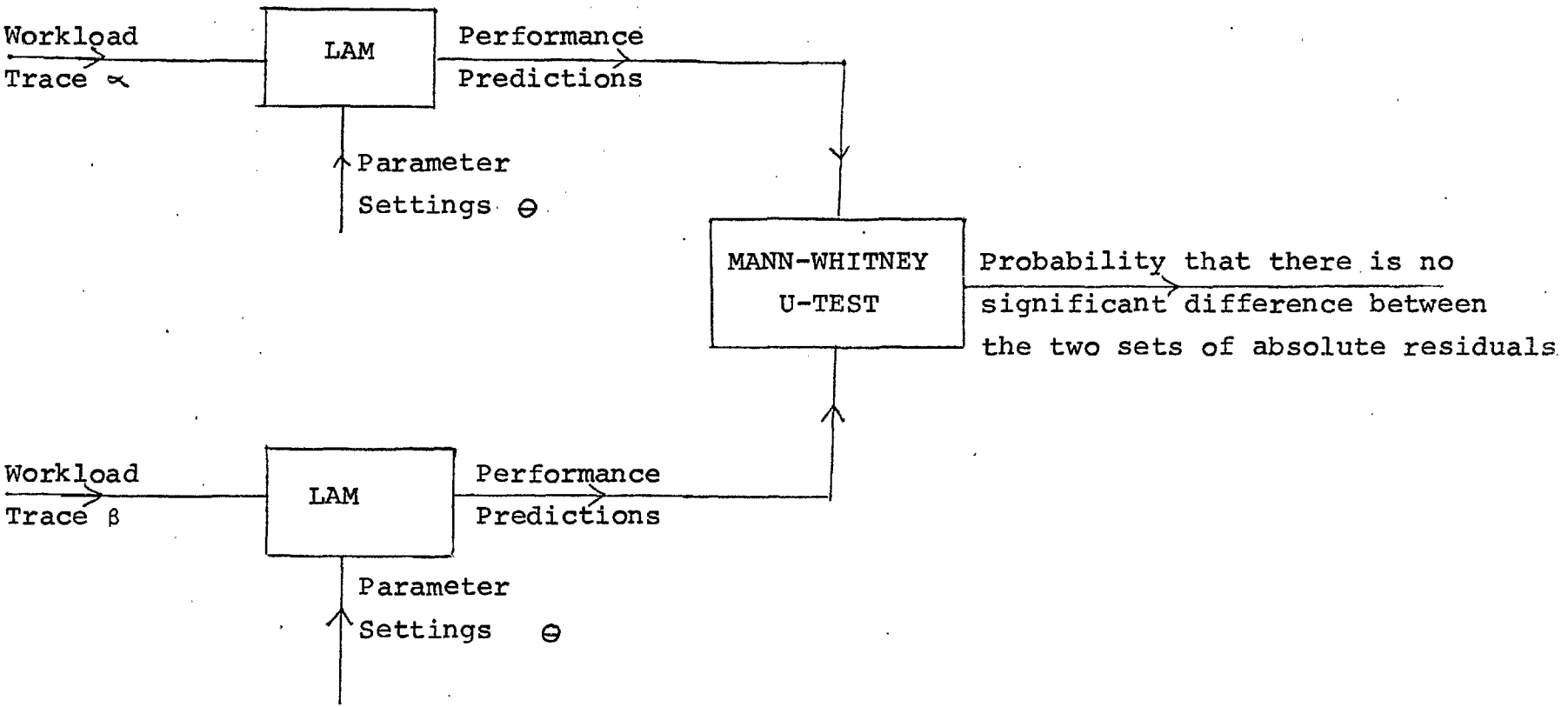


Figure 8.10: Procedure for Validating Load Adjusting Model

If the test rejects the null hypothesis, the validation is recommenced. Different parameter values, obtained from the calibration of the model with a different workload trace, are set. The model is then run with the other two traces respectively and the Mann-Whitney test is again applied to the sets of absolute residuals.

The validation was tried first using the parameter values obtained in the calibration of the 30/4 session (8.8.4). The Mann-Whitney test indicates the probability of the null hypothesis, that the two sets of absolute residuals have been drawn from the same distribution, being true. Table 8.3 shows that the null hypothesis may be accepted at the 10% level. Hence the LAM has been successfully validated for the three sessions under consideration.

8.10 Conclusions

This chapter has described the concepts of the hybrid Load Adjusting Model, which combines regression and simulation modelling techniques by creating a simulation framework which models job arrival and termination. By this means, an important structural limitation of the Workload Model, that the average load experienced by a job must be specified in advance of an experimental run, is overcome. Since the load will not usually be known in advance, the Load Adjusting Model is a more valuable model for experimental runs.

The viability of the approach has been shown by applying it to modelling the short job workload on the Imperial College Kronos system at the second level of detail. The parameters of the model were tuned during an iterative calibration procedure. The model was successfully validated for three separate sessions.

A limitation of the model is that the mean absolute value of the residuals is large. Table 8.3 shows that, for two of the sessions, the predictions of the validated LAM

Table 8.3: Validation of Load Adjusting Model

Parameter Settings

$b_o = 4.7$ $b_1 = 2.1$ $b_2 = 0.8$

$a_1 = 0.11$ $a_2 = 0.01$

Model Predictions

Session Modelled	actual $\overline{t_e}$ (secs)	predicted $\overline{t_e}$ (secs)	$\lvert\overline{r_r}\rvert$	$\lvert\overline{r_s}\rvert$	P
27/1	17.4	15.6	5.08	4.56	0.161
30/4	17.0	15.7	4.72	4.55	0.064
30/1	16.8	15.7	4.42	4.01	0.119

(For key refer to table 8.1)

Pairwise Comparison of LAM Predictions using Mann-Whitney U-Test

Session A	Session B	P^1
27/1	30/4	0.140
27/1	30/1	0.492
30/4	30/1	0.161

P^1 : Probability that there is no difference between the two sets of absolute residuals.

are not significantly better than the Workload Model. For the third, the predictions are significantly better at the 10% level. This limitation is mainly due to the fact that the data used by LAM is at the same level of detail as WM, although in one case different in nature. The average load experienced by a job is input to WM (figure 8.1), whereas it is predicted by LAM (figure 8.2) given the times of job arrival. This leads to certain situations where the predictions of WM are superior to LAM. This happens when a build-up of short jobs on the real system results from a perturbation which is not recorded in the data and hence cannot be modelled.

Most of the limitations of the Workload Model are due to a lack of data (7.10), and these are also reflected in the regression submodel. Data on jobs I/O demands and rollout time would enable a much more accurate regression model of execution time to be constructed.

The Load Adjusting Model, like the Workload Model, does not distinguish between short jobs executing in Central Memory and those rolled out. Furthermore, LAM is only capable of modelling the short job workload. Methods of overcoming both these limitations are discussed in the next chapter.

In spite of these limitations, it is believed that the Load Adjusting Model has clearly shown that, by overcoming the structural limitations of the purely regression model, the hybrid simulation/regression modelling approach is a valuable method of computer system modelling.

CHAPTER 9: THE MEMORY MANAGEMENT MODEL

9.1 Introduction

This chapter considers the limitations of the Load Adjusting Model and attempts to overcome one of the most important limitations by simulating the memory management subsystem. The Memory Management Model has been applied to the modelling of the Imperial College Kronos system at the third level of detail.

Section 9.2 discusses an important limitation of the Load Adjusting Model and how it may be overcome by developing the Memory Management Model. Section 9.3 describes the memory management subsystem in Kronos. Section 9.4 describes the assumptions made by the model and presents an overall description of the model. Section 9.5 describes the implementation and initial calibration of the model. Limitations in the model were revealed which led to the redesign of the model, as discussed in section 9.6. Section 9.7 describes the design of the Mark 2 model. Finally the calibration and validation of the model, which was similar to the method used for the Load Adjusting Model, are described in section 9.8.

9.2 Limitations of the Load Adjusting Model

In the Workload Model, a measure of the level of competition is given by the average number of short jobs competing for resources with a given job. This measure is input to the model as an input variable (figure 8.1). In the Load Adjusting model, the level of competition is estimated using a simulation framework. The estimate is based on the number of short jobs started, input as a trace, and the number of short jobs terminated, which is predicted by the model (figure 8.2). However, the level of competition, whether measured in level 1 or estimated in level 2, does not distinguish between jobs in Central Memory (CM) competing for the CPU and I/O, and jobs rolled out of CM. This is a clear limitation of these models.

The Memory Management Model attempts to estimate more precisely the load on the system at any given time, by simulating the memory management subsystem. This subsystem is modelled at a greater level of detail than the rest of the system. As with the Load Adjusting Model, the regression modelling approach is used to estimate a job's elapsed time in the absence of competition from other jobs. This minimum elapsed time is termed the job execution time. The time a job spends in the execution phase may be extended by competition from other jobs in CM for scarce resources, or by the job being rolled out of CM.

The delay, due to the competition with other jobs in the execution phase for scarce system resources, is estimated using the same method as in the level 2 model, that is by estimating the delay experienced by each job on a time segment basis. However, one important difference is that the predicted delay in the level 3 model is based on the estimated number of jobs resident in CM, rather than the estimated number of jobs in execution (as in level 2), some of which may be rolled out. For this to be possible, it is necessary to simulate the memory management subsystem.

9.3 Memory Management in Kronos

Memory management in Kronos is handled by:

- The Job Scheduler PP program, which mades scheduling recommendations to
- Monitor (the Nucleus of the Kronos Operating System) which allocates and de-allocates memory to and from the control points at which jobs run. Monitor bases its decisions on the Job Scheduler's recommendations.

Memory is allocated to a job at the job step level and occasionally at the sub-job step level. Each user specifies the maximum memory his job will require on the job card. For some job steps (e.g. any job step requiring the relocatable binary loader), the maximum memory allotment is allocated to the job. For system utility job steps, only the memory required by the appropriate system program is allocated.

When a job first enters the system, it is placed in the Input Queue. A job rolled out of CM is placed in the Rollout Queue. For the purpose of memory allocation, the Job Scheduler treats the Input and Rollout queues as one queue. It bases its decisions on:

(a) the job's CM priority - this is a priority associated with each job

(b) the amount of memory requested by the job

(c) memory availability and the CM priorities of the jobs resident in CM.

When a job enters the Input or Rollout Queue, its CM priority is set to an initial value. Its priority is gradually aged till an upper bound is reached. On the I.C. Kronos system, short jobs are given an initial priority which is above the upper bound. Consequently their priorities are not aged, and they are given preferential treatment over other batch jobs in the allocation of CM.

A job resident in CM is liable to be rolled out if a higher CM priority job makes a memory request. The higher CM priority job may:

(a) have entered the Input or Rollout Queue with a higher priority than a job in CM

(b) have had its priority aged past the priority of a job in CM.

(c) be resident in CM and have made a request for more memory.

Each job resident in CM is also awarded two time slices, a CPU time slice and a CM time slice. The CPU time slice is the amount of CPU time a job may use before becoming eligible for rollout. The CM time slice is the amount of real time a job may be resident in CM for, before becoming eligible for rollout. Usually, when a job becomes eligible for rollout, its CM priority is reduced, making it more likely that the job will be rolled out. However,on the I.C. system, the CM priority of short jobs is not reduced when a time slice expires; although eligible short jobs are liable to be rolled out if other short jobs are requesting memory.

9.4 The Memory Management Model of the Kronos System

9.4.1 Assumptions made by the Memory Management Model

A number of simplifying assumptions have been made in constructing the Memory Management Model. These are:

1. The only information on memory allocation that may be derived from the Dayfile is the average CM used by a job during its execution. Consequently, the model assumes that a job uses its average memory size throughout the execution phase. This is an obvious limitation to the model. It would be a simple extension to the model to handle memory allocation at the job step level if the data was available.

2. The model only models the short job workload. Since all short jobs have the same priority, no

priority scheduling is built into the model. It would again be a simple extension to the model to handle jobs of different priority.

3. Since all real time data derived from the Dayfile (e.g. job start and end times) are measured in units of a second, the basic real time unit in the model is the second.

4. Because of assumption 3 and because no measures of rollin/rollout time are available, the time taken to roll a job out and back in again must be neglected.

5. It is assumed that in each time segment, all jobs resident in CM are treated identically by the system. This is a reasonable assumption for CPU allocation, where a round robin scheduling algorithm is enforced. It is likely to be less reasonable for I/O management.

6. As a result of a local modification to the Account Dayfile in late 1974, the time a job was read through the card reader is no longer recorded. Consequently, the first message relating to a given job is recorded when the job starts execution. This time is used in the workload trace to represent job arrival in the system. This means that any job which is not scheduled immediately for execution by the model, and which enters the Input Queue instead, starts simulated execution later than it did on the real system.

These assumptions are bound to lead to inaccuracies in the model. Attempts are made to minimise these during the calibration of the model.

9.4.2 Overall Design of the Memory Management Model (MMM)

9.4.2.1 Job Commencement

When a job arrives in the system, at the simulated time of arrival, its execution time, (i.e. elapsed time in the absence of competition) is predicted by the regression submodel (see 8.3). The model then determines whether sufficient memory is available for the job, rolling out 'eligible' jobs if necessary.

If sufficient memory is available, an entry is set up for the job and is linked onto the end of the Execution List. This list contains an entry for each simulated job executing in CM. If insufficient memory is available, an entry is set up for the job and linked onto the end of a combined Input/Rollout Queue. This queue contains an entry for each simulated job in input or rollout state.

9.4.2.2 Time Slice Expiry

Each time round its main loop, the model checks each job in the Execution List to determine whether it has exceeded the CPU or CM time slice. The length of the CPU time slice is a system parameter and is currently set to four seconds. The model converts this into a real time measure, which is called the Execution time slice. At the simulated time of job commencement, the Execution time slice is estimated for each job whose total CPU time is greater than the CPU time slice:

$$\text{Execution time slice} = \frac{\text{Estimated Job Execution Time} \times \text{CPU time slice}}{\text{Total CPU time required}}$$

In estimating the Execution time slice, it is again assumed that a job uses the resources it requests uniformly. The CM time slice is a real time measure, currently set to 200 seconds, and so no conversion is necessary. If the

model determines that a job has exceeded either time slice, the job's state is set to 'eligible for rollout'.

9.4.2.3 Rollout Jobs

If insufficient free memory is available for a job at the simulated time of its arrival, then jobs in 'eligible for rollout state' are liable to be rolled out. Furthermore, each time round its main loop, the model checks if sufficient memory is available for one or more jobs in the Input/Rollout Queue to be rolled in. If necessary, jobs in 'eligible for rollout state' will be rolled out. A rolled out job is taken off the Execution list and placed at the end of the Input/Rollout Queue, and its state is set to rollout state.

9.4.2.4 Rollin Jobs

In the Kronos system, the Job Scheduler scans the File Name Table in one pass, to select the highest priority jobs in either Input or Rollout state that will fit into memory (after rollouts if necessary). In the model, a combined Input/Rollout Queue is maintained. Incoming jobs are placed on the end of the queue. The model attempts to fit each job in turn in the available memory (after rollouts if necessary). A job that can be allocated memory is taken off the Input/Rollout Queue and transferred to the end of the Execution list. Its state is changed to execution state.

9.5 Implementation and Initial Calibration

9.5.1 Implementation

The implementation approach to the Memory Management Model was similar to that adopted for the Load Adjusting Model (8.6). A preprocessor uses the B and J files for a given session to prepare a workload trace, which is input to the model. The model outputs a number of statistics

for each job, which are analysed by the Postprocessor. The predictions for each job are:

(a) Predicted job elapsed time.

(b) Predicted job execution time.

(c) Predicted delay time a job experiences due to competition from other jobs in CM.

(d) Predicted rollout time.

(e) Predicted time spent in Input Q.

The elapsed time predicted by the Workload Model for the same job is also output.

9.5.2 Problems in Calibrating the Memory Management Model

A number of problems exist in the calibration of the Memory Management Model (MMM). These are mainly due to the lack of performance data available to assist in the calibration.

As pointed out in section 3.4.3, in a multilevel modelling approach, the quantity and accuracy of the data input to the model should increase as the level of detailed representation increases. Although the level of detail was increased from level 1 to level 2, the quantity of data input to the model was different, though not greater in detail. Whereas, the average load experienced by each job is input to the level 1 Workload Model, this figure is predicted by the level 2 Load Adjusting Model, given the times of job arrival.

Increasing the level of detail from level 2 to level 3, more data is input to the model, but the increase is limited. The additional information is:

(i) the total amount of Central Memory available for user jobs, 50K 60 bit words

(ii) the average CM used by each job.

The main problems in calibrating this level are:

(a) because the average CM allocated to jobs is used, the model considerably underestimates the rate of change of memory allocation.

(b) No indication of rollin/rollout is given in the Dayfile. Thus no indication is given of which jobs were rolled out, why they were rolled out, and for how long.

The calibration process attempts to reduce the difference in behaviour between the real and modelled worlds. If the behaviour of the real world is not known in sufficient detail, this will considerably restrict the calibration process.

A trace driven simulation model of a CDC 6000 system has been described (N3) which also used the Dayfile as the source of workload and performance data. For each job, the total rollout time and number of times the job was rolled out were extracted. This data was used as input to the model. However, as these values are dependent on the system load, it is believed that they should be predicted by the model, as is the case with MMM. If this performance data were available, it should be used to assist in the calibration, as will be discussed in chapter 10.

9.5.3 The Calibration Approach

Because of the problems described in the previous subsection, the calibration approach adopted for the

level 3 MMM was very similar to the level 2 LAM, in spite of the fact that the system is modelled at a greater level of detail. To calibrate the MMM, the calibration parameters of the execution time and delay time submodels were adjusted in an attempt to reduce the difference in behaviour between the real and modelled worlds, (see sections 8.7 and 8.8 in chapter 8). The parameters are:

(a) The parameters bo, b_1 and b_2 of the regression submodel for predicting job execution time

$$t_j = bo + b_1 T + b_2 N \qquad (1)$$

(b) The parameters a_1 and a_2 of the submodel for predicting the delay experienced by each job in a time segment:

$$t_{di} = a_1 N + a_2 N^2 \qquad (2)$$

In the MMM, N refers to the estimated number of jobs in Central Memory competing with a given job, rather than the total number of short jobs in execution, whether in CM or rolled out, as used by the LAM.

In the calibration, the figure of merit used is again the absolute value of residuals. The iterative tuning approach is again adopted with the Method of Good Balance and Wilcoxon test applied at each stage.

The same three sessions were used for the calibration and validation of MMM as for LAM.

9.5.4 Initial Calibration Results

The initial attempts at calibrating the model used the 27/1 session as a basis. The results showed that:

(a) The number of jobs rolled out or placed in the Input Q on arrival, was small. For the more satisfactory parameter settings, it did not exceed 5% of the sample. The predictions for the number of jobs rolled out are almost certainly much lower than the actual number rolled out. The use of average CM is likely to smooth out the flow of jobs in the model, leading to considerably less rollin/rollout activity than in the real system.

(b) The Wilcoxon Test showed that with the more satisfactory parameter settings, the difference between the Memory Management and Workload models, as measured by the distribution of the two sets of absolute residuals, was not a significant one. At no stage however was the MMM significantly better than the WM.

It was apparent from analysing the results that for a large proportion of simulated time, the MMM was behaving in a manner similar to the Load Adjusting model. Indeed, the most satisfactory parameter settings for MMM were close to those for LAM. This was regarded as an unsatisfactory state of affairs, as it was felt that the level 3 model should be capable of producing better results. Although the calibration had not been satisfactory, it was felt that the problem lay in the model itself, rather than with the calibration method. It was therefore decided to re-examine the structure of the model and attempt to improve it.

The first attempt at improving the model was in the method of checking whether a job had exceeded its time slice, thereby making it eligible for rollout. A simplification had been made in the design of the model, which meant that a check on time slice expiry was only carried out each time a job start or job termination event

occurred. This was changed, so that a time slice expiry could now be a separate event. This meant that the model should be more accurate at estimating when a job becomes eligible for rollout, which should have the result of increasing the number of rollouts. Incorporating this change in the model did increase the number of rollouts slightly. However, it did not have a significant effect on improving the predicted job elapsed time.

9.6 The Memory Management Model Mark 2

9.6.1 Modelling the Long Job Workload

It became apparent that a radical change to the structure of the Memory Management Model was necessary, if the model was to be significantly improved. One way in which the model could be improved is in the prediction of job delay time.

In the first and second level models, only the short job workload is modelled, and the presence of the long job workload is ignored. This is because no measure is available of rollin/rollout. Hence there is no means of distinguishing between long jobs in CM and those rolled out. It is likely that for most of the time, most long jobs in the execution phase are rolled out and therefore do not affect the progress of the higher priority short jobs. However, those long jobs executing in CM are likely to affect the progress of short jobs, because once in CM, no preferential treatment is shown by the CPU and I/O scheduling algorithms. Hence, ignoring the long job workload is bound to introduce additional errors into the models.

In the version of the Memory Management Model just described, long jobs are also ignored. However, when the memory management subsystem is simulated, it becomes

possible to take into account the effect of long jobs. This is because, at this level, it becomes possible to make reasonable estimates of how many long jobs are executing in CM and how many are rolled out. It was therefore decided that the long job workload should be modelled in addition to the short job workload.

The method adopted for modelling the long job workload is similar to that used for the short job workload. A regression submodel is used to predict long job execution time. The delay time submodel is used for estimating the delay experienced by each job (long or short) in a time segment. The simulation of the memory Management subsystem is extended to include long jobs.

One method of modelling the long job workload is to attempt to construct a regression submodel to predict long job execution times, and to calibrate this in the same manner as the regression submodel for the short job workload. However, as discussed in chapter 6, it is doubtful whether representative regression models of the long job workload can be constructed in the framework of this project.

The objective of introducing the long job workload into the model is therefore limited to making better estimates of the competition experienced by short jobs, rather than for predicting the elapsed time of long jobs. Consequently, it was decided to use the same regression submodel for predicting job execution times for both long and short jobs.

9.6.2 Assumptions made by Memory Management Model Mark 2

A number of simplifying assumptions are made in modelling the long job workload:

1. The regression submodel for execution time is used to predict the execution time of long jobs. The reason for this was discussed in the last subsection.

2. A long job is assumed to use its average memory requirement throughput. This is similar to and for the same reason as the assumption made for short jobs.

3. Only two priority levels for access to CM are assumed, one for short jobs and the other for long jobs.

   In the Kronos system, no distinction is made between the different classes of jobs once they are executing in CM. Short jobs have a constant access priority to CM which is higher than for long jobs. Long jobs enter the Input and Rollout queues with a given priority which is aged until an upper bound is reached.

   In the model, two sets of Input/Rollout queues are maintained, one for long jobs in addition to the one for short jobs. Each job entering input or rollout states is placed on the end of the appropriate queue. The approximation is therefore a reasonable one.

4. As long jobs have a lower CM priority than short jobs, they are liable to be rolled out much more frequently. Consequently, no check is made on the time slice expiry of long jobs. The only effect of including this check might be to rollout one long job to allow another to execute.

5. In Kronos, no distinction is made between the different classes of jobs for CPU and I/O schedul-

ing. Therefore, it is assumed in the model that in each time segment, all jobs resident in CM are treated identically. Consequently, the same method as before is employed for estimating the delay time in each segment.

6. Since magnetic tape mounting time is not known for those jobs that use tapes, it is assumed that tape mounting is instantaneous. As the objective of modelling the long job workload is not for predicting long job delay time, this assumption is acceptable.

The assumptions made in modelling the long job workloads, many of which are cruder than those for the short job workload, are bound to lead to inaccuracies in the model. Nevertheless it was felt that modelling the long job workload, even in this crude form, was better than not modelling it at all, and that the result would be an improvement in the predictions for the short job workload.

9.6.3 Overview of Memory Management Model Mark 2

9.6.3.1 Introduction

This section provides an overview of the Memory Management Model Mark 2. The lists maintained by the model are increased by one. In addition to the Execution List, and combined Input/Rollout Queue for short jobs, an Input/Rollout Queue for long jobs is also maintained.

9.6.3.2 Job Commencement

When a long job arrives in the system, at the simulated time of arrival, its execution time is predicted by the regression submodel (equation 1). If sufficient

memory is available, the job is linked onto the end of the Execution List. If not, it is placed onto the end of the combined Input/Rollout Queue for long jobs. This contains an entry for every long job in Input or Rollout state.

When a short job arrives in the system, the model checks if enough memory is available for the job. If there is, the job is linked onto the end of the Execution List. If not, the model checks if there is enough 'eligible' memory in addition to free memory, to allow the job to start execution. Eligible memory is that used by executing jobs which are eligible for rollout. This includes all long jobs and all short jobs which have exceeded their time slice (see 9.4.2.2). If there is then sufficient memory, as many 'eligible' jobs as necessary are rolled out, and the new job is linked onto the Execution List. If insufficient memory is available, the job is placed on the end of the short job Input/Rollout Queue.

9.6.3.3 Rollout Jobs

If insufficient memory is available when a short job enters the system, jobs in 'eligible for rollout state' are liable to be rolled out. Two passes are made through the Execution List. First, long jobs are rolled out. Next short jobs in eligible state are rolled out. The search stops as soon as enough memory has been released to satisfy a memory request. Long jobs rolled out are placed on the end of the long job Input/Rollout Queue. Short jobs rolled out are placed on the end of the short job Input/Rollout Queue.

9.6.3.4 Rollin Jobs

Each time round its main loop (at each event occurrence), the model checks if sufficient memory is available for one or more jobs in first the short job and then the long job Input/Rollout queues to be rolled in.

Short jobs in eligible state are liable to be rolled out to allow short jobs (but not long jobs) to be brought into CM.

Starting at the head of the list, a check is made on each job in the short job Input/Rollout queue to see if sufficient memory, including eligible memory, is available for it to be rolled in (if it was in rollout state) or start execution (if it was in input state). If so, the job is brought into CM and placed on the end of the Execution List, after rolling out eligible jobs if necessary.

Next, a similar check is made on the long job Input/Rollout Queue. A long job is only rolled in, if sufficient free memory is available.

9.7 Design of the Memory Management Model Mark 2

The level 3 Memory Management Model is a refinement of the level 2 Load Adjusting Model described in chapter 8. The design uses the same framework as the level 2 design. Some new routines have been added, others have been extended and the remainder have been left unchanged. A block diagram showing the overall design of the Memory Management model is shown in figure 9.1. A short description of each of the routines follows:

MAIN is the main routine. It carries out a necessary initialisation and then enters COORD.

COORD is the co-ordinating routine for the model. It goes through a main loop, once for each time segment. First it calls SENEVT to determine what the next event is, and to determine the length of the next time segment. It then calls UPDENT to update all job entries. If any jobs are due to terminate, the job termination routine JBTERM is

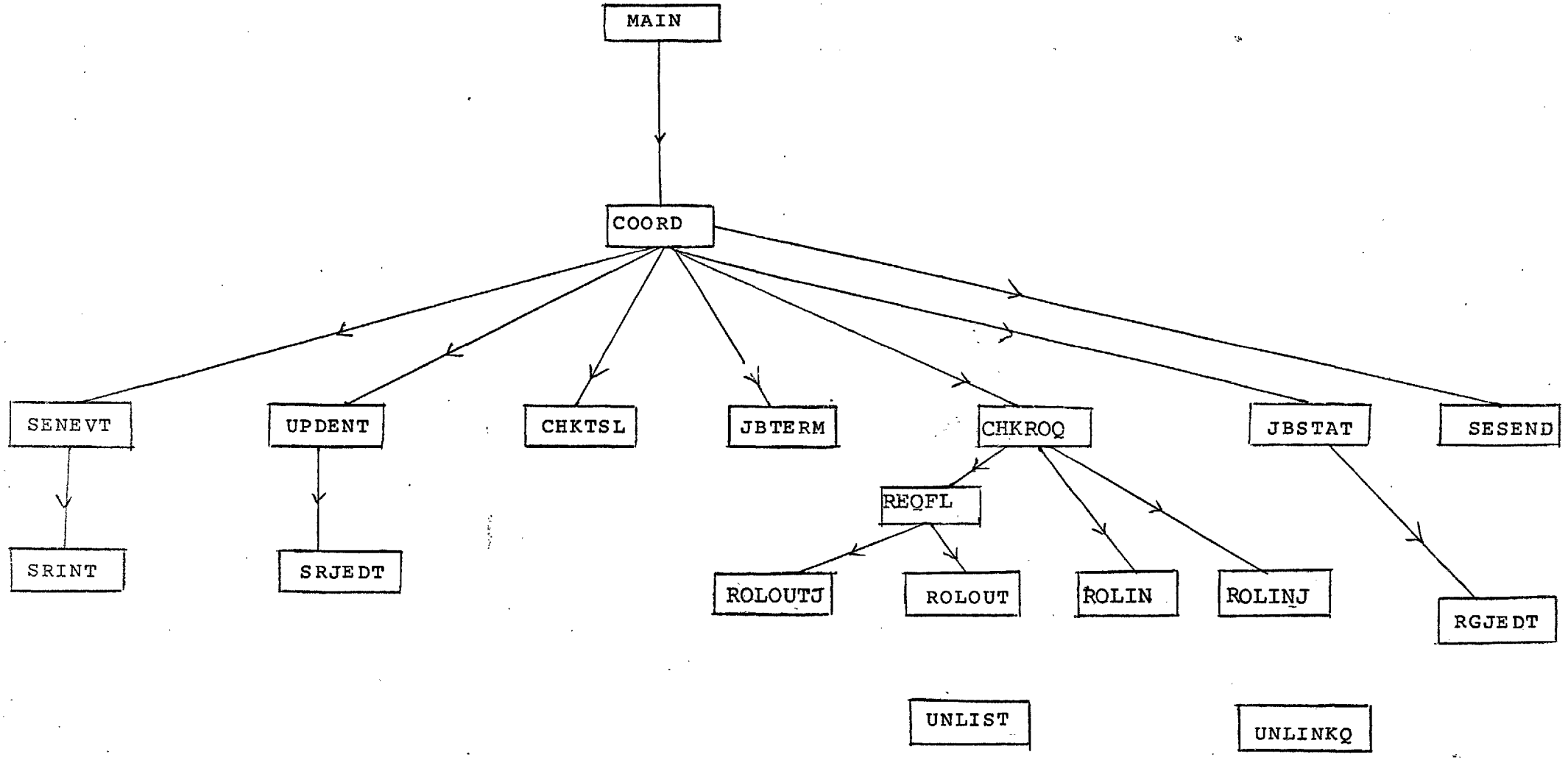


Figure 9.1: Structure of the Memory Management Model

called. Next, the routine CHKTSL is called to check whether any short jobs in executing state have exceeded a time slice. Next, the routine CHKROQ is called to check jobs in the Input/Rollout queues to determine if any jobs may be brought into CM. CHKROQ is called twice, first for short jobs, then for long jobs. Finally if a job is due to arrive in the system (as determined by the input trace), the job commencement routine JBSTAT is called.

SENEVT is called by COORD to determine what the next event is, and the estimated time at which it occurs. The next event could be a job arrival, job termination, time slice expiry, or end of session. SRINT is called to estimate what the minimum segment length for the next job to terminate is. This is compared with the next job arrival time and the time for the next time slice expiry event to decide what the next event is. If the number of jobs currently in execution is equal to a certain limit, then no jobs will be allowed to commence execution, until the number of jobs in execution falls below this limit.

UPDENT Given the length of the next time segment t_i, UPDENT calls SRJEDT to compute the delay time t_{di} and the execution time t_{ji} for the segment (chapter 8). The entries are updated for each job on the Execution list, as follows:

(a) the delay time for this interval is added to the cumulative delay so far.

(b) the execution time for this interval is decremented from the remaining execution time.

UPDENT also scans both the short job and long job Input/Rollout Queues and for each job in rollout state, the segment length is added to the rollout time so far.

JBTERM handles job termination. The entry for the job terminating is taken off the Execution list. The accumulated job statistics are copied into a job output buffer. If the buffer is full, it is output to a disc file.

JBSTAT handles job arrival. The job's resource requirements and other information are read from the job's entry in the input buffer and copied into an entry created for the job. RGJEDT is called to predict the job's execution time. If sufficient memory is available, the job is scheduled for execution and linked onto the end of the Execution list. Its state is set to executing. If insufficient memory is available, the job is linked onto the end of the appropriate Input/Rollout Queue and its state is set to input. Finally, JBSTAT checks if the input buffer is empty, and if so replenishes it.

CHKROQ is called by COORD to check the Input/Rollout Queues for either short jobs or long jobs. For the queue selected, CHKROQ starts with the first entry in the queue, extracts the memory request, and calls REQFL to determine if the request can be satisfied. If it can, then the job is brought into CM by calling ROLIN for short jobs or ROLINJ for long jobs. Each job in the appropriate queue is checked in this manner.

REQFL checks if sufficient free Central Memory is available to satisfy a memory request from either a short or long job. If sufficient free memory is

available, a positive response is returned. If not, and the job is a long job, a negative response is returned. For short jobs, a further check is made to determine if enough memory is available when the CM used by jobs eligible for rollout is taken into account. If enough memory is still not available, a negative response is returned. Otherwise, ROLOUTJ is called to rollout as many long jobs as necessary. If more memory is still required, ROLOUT is called to rollout as many short jobs in eligible state as necessary. A positive response is then returned.

ROLOUTJ handles the rolling out of long jobs. If no long job is in executing state, a return is made to the calling routine REQFL. Otherwise the Execution list is searched for long jobs. A long job is removed from the list and placed on the end of the long job Input/Rollout Queue. The free memory count is increased. If sufficient memory is available to satisfy the request, the search is stopped. Otherwise, it is continued in the same manner until the end of the list is reached.

ROLOUT handles the rolling out of short jobs. The Execution list is searched for short jobs in 'eligible for rollout' state. Each job in this state is removed from the Execution list and placed on the end of the short job Input/Rollout Queue. The free memory count is increased. When sufficient memory is available to satisfy the request, the search is stopped.

ROLIN brings a given short job into Central Memory. The job is unlinked off the short job Input/Rollout Queue and placed on the end of the Execution list. The job, which may have been in input or rollout states, is put in execution state.

ROLINJ brings a given long job into CM. The job is unlinked off the long job Input/Rollout Queue and placed on the end of the Execution list.

SESEND is called at the estimated time of the end of session. It outputs the final bufferful of data to the disc file, and outputs statistics of the run to the line printer.

UNLIST (N,A,B) searches list A for the entry for job N. When found, the entry is unlinked off list A and linked onto the end of list B.

UNLINKQ (N,A,B) unlinks the top entry off list A and links in onto the end of list B. The entry number is returned in N.

RGJEDT computes the predicted job execution time t_j for a job commencing execution (see chapter 8).

SRINT computes the estimated time required to complete execution of a given job for a given load on the system (see chapter 8).

SRJEDT computes the estimated job execution time t_{ji} and the estimated delay time t_{di} for a given time segment t_i and load N on the system (see chapter 8).

TRACE outputs a trace message each time it is called, providing the trace flag is on. The message is either for job commencement, job termination, job put in Input Queue, job rolled out of CM or rolled back in again.

ERROR (I) is the error routine which may be called from a number of places in the model. It outputs an error message, dumps the contents of various locations and arrays, and stops.

9.8 Calibration and Validation

9.8.1 Calibration

After the major changes to the Memory Management Model to incorporate the modelling of the long job workload, the calibration process was recommenced. The calibration of the model with individual traces of the three sessions, the 27/1, 30/1 and 30/4 followed on similar lines to the detailed example given in section 8.8.4.

An analysis of residuals revealed that at certain times for each session, large positive residuals were obtained. The cause for this was similar to that described in 8.8.3, and due to the difference in basic structure between the Workload and Memory Management models. The problem was resolved, as before, by excluding certain observations.

The calibration now showed that the Mark 2 Memory Management Model was a considerable improvement over the Mark 1 model, and to a lesser degree over LAM. An example of four runs with the various models using the 27/1 trace, is shown in table 9.1. The runs are:

(i) The calibrated LAM, with the 'best' parameter settings for 27/1.

(ii) MMM Mark 1 with the same parameter settings as (i).

(iii) The calibrated MMM Mark 2 with the 'best' parameter settings for 27/1.

(iv) MMM Mark 1 model with the same parameter settings as (iii).

The results show that the calibrated MMM Mark 2 model (run iii) is:

Table 9.1: Comparison of Models

Run	Model	b_o	b_1	b_2	a_1	a_2	predicted $\overline{t_e}$	$\lvert\overline{r_s}\rvert$	P
i	LAM	4.7	2.3	0.8	0.1	0.01	16.4	4.45	0.037
ii	MMM MK.1	4.7	2.3	0.8	0.1	0.01	16.1	4.57	0.125
iii	MMM MK.2	4.7	1.9	0.7	0.1	0.01	16.5	4.25	0.000
iv	MMM MK.1	4.7	1.9	0.7	0.1	0.01	14.1	4.90	0.488

Mean actual $\overline{t_e}$ = 17.4 seconds
Mean predicted $\overline{t_e}$ by WM = 18.3 seconds
$\lvert\overline{r_r}\rvert$ = 5.08

For key refer to table 9.2

Comparison of Models using Wilcoxon Test

Run A	Run B	P^1
iii	i	0.074
iii	ii	0.016
i	ii	0.030

P^1 : Probability that there is no difference between the two models.

(a) significantly better than the Workload Model at the 0.1% significance level (i.e. with 99.9% confidence).

(b) Significantly better than the calibrated LAM (run i) at the 10% level.

(c) Significantly better than either of the Mark 1 runs (ii and iv) at the 5% level.

The basic problem with the Mark 1 model is shown (table 9.1) by the fact that with the LAM parameter settings (run ii), the results are better than with the Mark 2 (run iv) parameters settings. The reason for this is that for a large proportion of simulated time, the Mark 1 model behaves in a manner similar to LAM.

9.8.2 Comparison of Sessions

The MMM Mark 2 was calibrated separately for the three different workload traces, where each trace represented a different session. The results are shown in table 9.2 and are now compared:

(a) The Wilcoxon test indicates that the predictions of the Memory Management Model are significantly better than the Workload Model at the 10% confidence level (i.e. with 90% confidence) for the 30/4 session, at the 5% level for the 30/1 session and at the 0.1% level for the 27/1 session.

(b) The parameter of the regression submodel, bo, b_1 and b_2 in equation 1, are identical for all three sessions. The differences in the parameter settings for the three sessions are in the delay parameters a_1 and a_2.

Table 9.2: Comparison of Modelled Sessions

Session Modelled	b_0	b_1	b_2	a_1	a_2	actual $\overline{t_e}$ (secs)	predicted $\overline{t_e}$ by MMM	$\lvert\overline{r_s}\rvert$	P
27/1	4.7	1.9	0.7	0.1	0.01	17.4	16.5	4.25	0.000
30/4	4.7	1.9	0.7	0.073	0.013	17.0	15.8	4.54	0.087
30/1	4.7	1.9	0.7	0.07	0.01	16.8	15.2	3.99	0.046

Key: $\overline{t_e}$: mean job elapsed time in seconds

$\lvert\overline{r_r}\rvert$: mean of absolute residuals of WM

$\lvert\overline{r_s}\rvert$: mean of absolute residuals of MMM

P : probability that there is no difference between MMM and WM

9.8.3 Validation

The same approach to the validation of MMM was adopted as for LAM (section 8.9). The model parameters are set to values obtained in the calibration of MMM with a given workload trace. The model is then run with the other two traces respectively. The Mann-Whitney U-test is used to determine if there is any significant difference in the model predictions. If the test fails, the validation is recommenced using a set of parameter values, obtained in the calibration of the model with a different workload trace.

The results of the validation are shown in table 9.3. The Mann-Whitney test shows that the null hypothesis, that two sets of absolute residuals have been drawn from the same population, may be accepted at the 4% level. Thus the MMM has been successfully validated at this level for the three sessions under consideration.

With the parameter settings of table 9.3, the predictions of MMM are significantly better at the 5% level than the WM for two sessions, 27/1 and 30/1. For the 30/4 session, there is no significant difference between the models.

9.9 Conclusions

This chapter has described the modelling of the I.C. Kronos system, at the third level of detail, with the simulation of the memory management subsystem. By this means, an important limitation of both the Workload and Load Adjusting Models (neither of these models distinguished between jobs executing in Central Memory and those rolled out) was overcome.

The calibration of this model suffered from a scarcity of performance data for calibration purposes. Initially

Table 9.3: Validation of Memory Management Model

Parameter Settings

$b_0 = 4.7 \quad b_1 = 1.9 \quad b_2 = 0.7$

$a_1 = 0.07 \quad a_2 = 0.01$

Model Predictions

Session Modelled	Actual $\overline{t_e}$(secs)	Predicted $\overline{t_e}$	$\lvert\overline{r_r}\rvert$	$\lvert\overline{r_s}\rvert$	P
27/1	17.4	15.4	5.08	4.51	0.037
30/4	17.0	15.7	4.72	4.58	0.133
30/1	16.8	15.2	4.42	3.99	0.046

For key refer to table 9.2

Pairwise Comparison of MMM Predictions Using Mann-Whitney U-Test

Session A	Session B	P'
27/1	30/4	0.057
27/1	30/1	0.444
30/4	30/1	0.042

P' : Probability that there is no difference between the two sets of absolute residuals.

only the short job workload was modelled, but this approach was not satisfactory. It was decided therefore to also include the long job workload in the model. However this could only be done in a very approximate manner, as no regression model of the long job workload existed. In spite of this, the approach resulted in a significantly better model. The Memory Management Model was calibrated and successfully validated for three separate sessions.

The limitations of the model are due mainly to the limitations of the workload and performance data available from the Dayfile:

(a) The average CM requested by each job is used in the model. This means that the model is bound to considerably underestimate the rate of change in memory allocation.

(b) No data is recorded on rollin/rollout activity whatsoever. This considerably restricts the calibration. Ways in which the presence of this data could help the calibration are discussed in chapter 10.

(c) Long jobs are modelled very approximately. The short job submodel is used to predict long job execution times.

In spite of these limitations, it is felt that the Memory Management Model has demonstrated further the viability and advantages of the multilevel hybrid approach to computer system modelling. The model builder need only model in detail that part of the system of particular interest, using simulation techniques. Other parts of the system may be modelled at a much less detailed level, using regression techniques.

CHAPTER 10: EVALUATION AND PROPOSALS FOR FUTURE WORK

10.1 Introduction

Three models of computer system performance have been constructed and applied to modelling the Imperial College system at three levels of detail. Section 10.2 evaluates the three models. Section 10.3 discusses methods of developing more accurate versions of the three models. Methods of modelling the system at a greater level of detail are discussed in section 10.4.

Although only applied to one system, the modelling approach developed in this thesis may be applied to other non-virtual storage batch systems. Further research is needed to extend the approach to modelling virtual storage systems. Methods of doing this are suggested in section 10.5.

10.2 Evaluation of the Models

10.2.1 Main Results

The main results of the research described in this thesis are:

(a) Validated regression models of computer system performance may be developed by applying a systematic approach to the evaluation.

The Workload Model is a purely regression model of the short job workload. The model was validated for four different sessions, covering a period of four months. In other words, it was shown that a single regression equation is adequate to explain each of the four data samples.

Regression analysis may initially appear to be a simple method of computer system modelling, but in practice this was not found to be so. The analysis

described in this thesis has shown that to build meaningful and consistent models of computer system performance requires a very comprehensive analysis of the characteristics of the workload and of the residuals of the model, coupled with a thorough understanding of system behaviour. It was revealing to discover that the process of developing the model actually helped explain certain aspects of system behaviour. Understanding system behaviour was essential for explaining the inadequacies of earlier versions of the model, for assisting in decision-making and for interpreting the regression coefficients of the final model.

(b) Regression and simulation modelling techniques may be combined to construct a hybrid model, which is a dynamic model of system performance.

To model variations in load in a regression model, it is necessary for one or more of the independent variables to be measures of system load. This limitation is overcome in the hybrid Load Adjusting Model by the creation of a simulation framework which models job arrival and termination. By this means, LAM is capable of dynamically adjusting its estimate of the load on the system. Hence, the LAM is more valuable than the Workload Model for experimental purposes. The LAM was successfully calibrated and validated for three different sessions, demonstrating the feasibility and value of this approach.

(c) The hybrid simulation/regression framework provides a very useful means of modelling the system at different levels of detail. Those parts of the system of special interest may be modelled at as detailed a level as required using simulation techniques. The rest of the system may be modelled at a much less detailed level using regression techniques.

The approach has been demonstrated by modelling the Imperial College system at three levels of detail, the second and third of which make use of the hybrid framework. The Memory Management Model is derived from the Load Adjusting Model by a systematic expansion of detail. It is a hybrid model in which the memory management subsystem is modelled at a much greater level of detail than the rest of the system. The model was successfully calibrated and validated for three different sessions.

10.2.2 Comparison of the Models

Table 10.1 shows the relative speeds of the three models on the CYBER system. It shows the compilation times and the execution times required to model one $3\frac{1}{2}$ hour session on the CYBER. The results show that the models are extremely fast. Thus, experimenting with any of these models is more than three orders of magnitude more economical than experimenting with the real system.

Table 10.2 compares the predictions of the three validated models for each of the three sessions. The following results are displayed:

(a) The mean of the actual job elapsed time, i.e. as measured on the CYBER.

(b) Mean of the absolute value of the residuals for each model.

(c) (b) as a percentage of (a) for each model. This is a measure of the accuracy of the predictions.

(d) The result of the Wilcoxon test, i.e. the probability that the predictions of one model are not significantly better than another model, for the same session.

Table 10.1: Comparison of Model Speeds

	Compilation Time (secs)	Execution Time * (secs)
Workload Model	1.8	0.7
Load Adjusting Model	2.3	3.4
Memory Management Model	4.6	4.6

* Time to process 3½ hour session on 27/1/75

Table 10.2: Comparison of Model Predictions

Session	Measured	Workload Model		Load Adjusting Model			Memory Management Model			
	$\overline{t_e}$ (secs)	$\lvert\overline{r}\rvert$ (secs)	$\frac{\lvert\overline{r}\rvert}{\overline{t_e}}$ (%)	$\lvert\overline{r}\rvert$ (secs)	$\frac{\lvert\overline{r}\rvert}{\overline{t_e}}$ (%)	P_1	$\lvert\overline{r}\rvert$ (secs)	$\frac{\lvert\overline{r}\rvert}{\overline{t_e}}$ (%)	P_2	P_3
27/1	17.4	4.93	28.3	4.56	26.2	0.351	4.51	25.9	0.087	0.181
30/4	17.0	4.61	27.2	4.55	26.8	0.185	4.58	26.9	0.380	0.344
30/1	16.8	4.56	27.1	4.01	23.9	0.069	3.99	23.8	0.023	0.419

Key: $\overline{t_e}$: Mean actual elapsed time

$\lvert\overline{r}\rvert$: Mean absolute value of residuals for a given model

P_1 : Probability that there is no significant difference between LAM and WM

P_2 : Probability that there is no significant difference between MMM and WM

P_3 : Probability that there is no significant difference between MMM and LAM

Table 10.2 shows that:

(a) The mean absolute error in the predictions as a percentage of mean actual elapsed time is between 27% and 28% for the Workload Model, and between 24% and 27% for both the Load Adjusting and Memory Management Models.

(b) The predictions of the Load Adjusting Model are significantly better than the Workload Model at the 10% level in one case out of three.

(c) The predictions of the Memory Management Model are significantly better than the Workload Model at the 10% level in two cases out of three.

(d) In none of the cases are the predictions of the Memory Management Model significantly better than the Load Adjusting Model.

The results show that the residuals of all three models are comparatively large. The main reason for this is the limitations of the available data. As pointed out in section 3.4.3, the amount of workload data collected for input to the model and performance data for calibration purposes should increase as the level of detail of the model increases. However in the modelling of the Kronos system, the quantity and accuracy of the data, all derived from the Kronos Dayfile, were basically unchanged for all three models. Whereas the Workload Model uses the average load experienced by each job as input, this figure is predicted by both the Load Adjusting and Memory Management Models, given the times of job arrival. The MMM also uses the average memory requirement of each job and the total amount of Central Memory available to user jobs.

Furthermore, as has already been pointed out, the data collected by the Dayfile has some severe limitations. The results of this research indicate strongly that if more workload and performance data were available, the models could be significantly improved (see 10.3).

10.2.3 The Problems of Calibration and Validation

This section discusses the different approaches to the calibration and validation of the regression and hybrid models. The Workload Model is a least squares regression equation. The objective of the least squares method is to minimise the sum of the residuals squared. Hence, as measured by this figure of merit, the parameters (i.e. intercept and regression coefficients) of the Workload Model are optimum.

The calibration of the hybrid models uses an iterative tuning approach. The Method of Good Balance is a very valuable means of guiding the tuning process. However, it is possible for the model to be well balanced while still capable of further improvement (see 8.8.4). From this stage on, the choice of parameter settings is intuitive and the tuning process becomes considerably more difficult. Calibration is stopped when no further improvement in the model is obtained.

In a regression model, the mean of the residuals is zero, that is the means of the observed and predicted values of the dependent variable are equal. However, in the hybrid models, the tuning process leads to a biased model, that is one where the model underestimates the job elapsed time. This is a consequence of the tuning process only considering the absolute value, and not the sign, of the residuals.

The calibration process was by far the most time consuming task in the implementation of the two hybrid models. There is no doubt that there is considerable scope for further research into developing more sophisticated statistical methods of calibrating and validating computer system models.

10.3 Extending the Models

10.3.1 Improving the Workload Model

This section discusses methods of improving and extending the Workload, Load Adjusting and Memory Management Models.

A method of improving all three models would be to improve the predictions of the regression Workload Model. This would also result in improvements to both the Load Adjusting and Memory Management Models, as both use a regression submodel.

Three areas where the Workload Model is in need of improvement are:

(a) There is no I/O term in the model. This is because the available measures of I/O demand are inadequate. Hence, the predictions of the model are poor for those jobs whose I/O demands differ greatly from the average. Better measures of I/O demand would be

  (i) non-overlapped I/O time
  (ii) number of I/O requests

(b) No measures of job rollout time are available. There is substantial evidence to support the view that if rollout time were accounted for in the model, the model would be significantly improved:

  (i) It was shown that when the system was heavily loaded and hence rollin/rollout activity was high, the regression models of the short job workload were poor (7.3). When these periods were excluded, much better models were built.

  (ii) When the batch workload was modelled in the absence of the timesharing load (7.7), thereby eliminating one major cause of short job rollout, the models were much more satisfactory.

  (iii) When the subset which did not experience any competition from other short jobs, (and hence should never have been rolled out) was modelled (7.9), the standard error of the residuals was considerably reduced.

(c) The measure of short job competition used in the model does not distinguish between short jobs in Central Memory competing for resources and those rolled out. The approximate nature of this measure is reflected in the greater fluctuation of the regression coefficient of this variable (7.7). A better measure of short job competition would be the average multiprogramming level.

Given this additional data, a regression model could be constructed in which the dependent variable would be the job memory residence time (i.e. elapsed time - rollout time). Independent variables would be the measures of resource demand and multiprogramming level. With the elimination of the uncertainty of rollout times and improved measures of I/O time and system load, these are strong grounds for believing that this model should be a significant improvement over the Workload Model.

Furthermore, a model of this type is likely to be more satisfactory for modelling the long job workload, as long jobs may be rolled out for substantial periods of time when the short job load is high.

A separate method would now be necessary for predicting rollout time. Regression techniques could be tried, although it is uncertain how successful this approach would be. A more satisfactory method is probably that of simulating the memory management subsystem as in the Memory Management Model.

10.3.2 Regression Models at the Job Step Level

More detailed regression models could be developed by modelling the system at the job step level. For this to be possible, all measurements of job resource demands and all performance measures would need to be collected at the job step level instead of the job level.

Different models could be built for different types of job steps, e.g. compilation, link loading, execution etc. (see section 3.5.3). In these models, the dependent variable would be the job step elapsed time. Independent variables would be the job step resource demands and measures of the load on the system. For a given job, the predicted elapsed time is the sum of the predicted job step elapsed times.

10.3.3 Regression Models of the Timesharing Workload

If more performance data were available, the timesharing workload could also be modelled. An appropriate performance measure for a timesharing environment is the response time, that is the time from when a transaction is requested by the user to the time the user starts receiving the response. The response time will depend on the resources required to process the transaction and on the load on the system.

To model the system at the transaction level, performance data at this level is necessary. Regression models could be constructed in which the dependent variable is the response time. Independent variables would be:

(i) Measures of the transaction resource demands.
(ii) Measures of the load on the system.

10.3.4 Extending the Load Adjusting Model

The Load Adjusting Model could be improved by:

(a) developing a more accurate regression model of job execution time. This could be done by using the regression model of job memory residence time proposed in 10.3.1 as a basis.

(b) Modelling the system at the job step level. This would necessitate developing regression submodels at this level. Job step, as well as job, start and termination events would need to be simulated.

The LAM could be extended to model the whole workload in a system which did not have any rollin/rollout activity, such as OS/360 MVT (multiprogramming with a variable number of tasks). For any system with rollin/-rollout activity, a more accurate dynamic model of the whole workload could be developed if the memory management subsystem was modelled.

10.3.5 Extending the Memory Management Model

The performance data available for calibrating the model was limited. More accurate versions of the model could be constructed if more performance data was available. In particular the following steps could be undertaken:

(a) A more accurate regression submodel of job execution time, as described for LAM, could be developed.

(b) Job rollout time is necessary for a more accurate calibration. Real and predicted rollout times could then be compared. Attempts could be made to reduce the difference between them.

(c) A separate regression submodel of long job execution time could be developed. This would enable the long job workload to be modelled more accurately.

(d) Regression submodels could be developed at the job step level. Several regression submodels of job step execution time, reflecting different characteristics of the workload, could be developed as described in 3.5.3 and 10.3.2.

(e) Job arrival time in the Input Queue should be recorded in the Dayfile. This would enable job scheduling to be modelled more accurately.

(f) System job activity should be recorded to allow system jobs to be modelled.

(g) Measurements of the time taken to roll jobs in and out of CM should also be available. This aspect of memory management could then be included in the model.

(h) MMM could be extended to model the timesharing load. A regression submodel of transaction execution time could be developed in the same way as proposed for the batch job step execution submodel.

10.4 Modelling at a Greater Level of Detail

The multilevel hybrid modelling approach could be refined further by modelling the system at greater levels of detail. For example, the time spent by a job resident in main memory could be modelled in more detail by modelling CPU and I/O management. A number of different approaches could be tried. In general, the more detail included in the model, the more workload and performance data required.

(a) A queuing submodel(s) is constructed to predict CPU and I/O utilisation for jobs resident in main memory (K3).

(b) The CPU and I/O scheduling algorithms are simulated. It is assumed that each job uses the average CPU and I/O burst throughput (W2).

(c) Increasing the accuracy of (b) by collecting a detailed event trace for every job in which the length of each CPU and I/O burst is measured (S4) and used as input to the model (S7).

(d) A submodel could be constructed for each I/O device, whose function is to predict I/O transfer times, given the characteristics of the device and data

about each transaction. Attempts could be made to construct these submodels using queuing, regression or simulation techniques.

10.5 Modelling Virtual Storage Systems

10.5.1 Introduction

The multilevel hybrid modelling approach may be applied to other non-virtual storage batch systems, providing sufficient data is available. Further research is necessary to investigate how this approach could be applied to virtual storage systems. Some suggestions are made in this section.

Because of the increased complexity in virtual storage systems, modelling these systems is also more complex. A number of analytical models of virtual storage systems have been developed (B8, B9, C5, C6, D2). The objective of developing these models, however, has usually been to study properties of virtual storage systems, rather than to relate the model predictions to actual computer system performance. Some simulation models of virtual storage systems have been developed (B10, N1, W6), but these are often very detailed models which are considerably slower than the system itself.

10.5.2 Hybrid Models of Virtual Storage Systems

There are a number of problems to be overcome in constructing hybrid models of virtual storage systems. A major problem is that of predicting paging rates, as the relationship between program behaviour, multiprogramming and paging is very complex.

It is proposed first to construct regression models in which job elapsed time is the dependent variable, as before. Independent variables may again be in one of two categories, resource demands, or measures of load on the system. The measures of load on the system should now include paging rates. Once constructed, such a model would, as before, provide the basis for a regression submodel within a hybrid model.

A possible approach to constructing a hybrid model might be to start with a model similar in concept to the Memory Management Model, in which each job's memory requirement is given by its estimated working set size (D2). Some method is then necessary for predicting the delay time due to paging.

One method of predicting a program's paging rate is by means of the parachor curve (M3). A parachor curve is a graph of the total number of page interrupts a program encounters as a function of the amount of physical memory available for holding pages. It would be necessary to develop a parachor curve, or more realistically a set of parachor curves, for the workload on a given system (M3). The paging delay experienced by each job could then be estimated using the appropriate curve.

A possible alternative approach might be to obtain first, by performance monitoring, a page trace of every program executed (B3). It is then proposed that a submodel be constructed which predicts the occurrence of page interrupts given the program trace, the program's CPU and I/O requirements, and the amount of physical memory available to it. It should then be possible to simulate the memory and paging algorithms.

Other possibilities might include the use of analytical submodels for predicting paging rates.

CHAPTER 11: CONCLUSIONS

This thesis has investigated some aspects of developing fast approximate models of computer system performance. Two different modelling techniques, regression and simulation modelling, have been applied and a method developed for combining their use within a multilevel hybrid modelling framework. The main objective of this thesis has been to demonstrate the feasibility and value of this approach to the modelling and evaluation of computer system performance.

The approach has been demonstrated by modelling the short job workload on the Imperial College CDC 6000 Kronos system, at three levels of detail. At each level, a self-contained model of the system has been developed. At the first level, the Workload Model uses purely regression techniques. At the second level, the Load Adjusting Model, simulation techniques are introduced and combined with the regression techniques. At the third level, the Memory Management Model, more detail is introduced with the simulation of the memory management subsystem.

All the workload and performance data, used for input to and calibration of the models, was extracted from the Kronos accounting subsystem called the Dayfile. The performance data collected by the Dayfile is limited. This has proved to be the major limitation in the development of the models.

The Workload Model is a purely regression model of computer system performance. It was developed after a comprehensive performance analysis and analysis of residuals. The model expresses a batch job's elapsed time as a function of the job's resource demands and load on the system. The model has been successfully validated for four different sessions.

One disadvantage of the Workload Model is that it is a static model and hence is not capable of dynamically estimating the load on the system. The hybrid Load Adjusting Model overcomes this limitation by creating a simulation framework within which regression techniques may be used. By this means, the model dynamically adjusts its estimate of system load as each simulated job executes. The Load Adjusting Model has been successfully calibrated and validated for three different sessions.

A disadvantage of both the Workload and Load Adjusting Models is that neither distinguish between jobs resident in Central Memory and jobs rolled out to secondary storage. The Memory Management Model overcomes this limitation by simulating the memory management subsystem. A more accurate estimate of the load on the system is made possible by also modelling the long job workload. The Memory Management Model has been successfully calibrated and validated for three different sessions.

By successfully constructing, calibrating and validating the three models of the Imperial College system, the feasibility and advantages of combining regression and simulation modelling techniques within a multilevel modelling framework have been demonstrated. By this means the advantages of both techniques are exploited. Regression analysis provides a fast quantitative method of modelling a system or subsystem at a gross level. Simulation models the passage of time and provides a means of modelling the system in more detail by representing logical and structural relationships in the system.

Although the modelling approach described in this thesis has been applied to only one system, the approach may be applied to other non-virtual storage batch systems. Methods for improving the three models further and for extending the approach to model the system at greater levels of detail have been proposed. Methods of extending the approach to virtual storage systems have also been suggested.

REFERENCES

B1 Y. Bard, 'Performance Criteria and Measurement for a Time Sharing System', IBM Systems Journal, Vol. 10, No. 3, 1971.

B2 Y. Bard and K.R. Suryanarayana, 'On the Structure of CP Overhead', page 329 in Reference F2.

B3 M. Baylis, D. Fletcher and D.J. Howarth, 'Paging Studies made on the I.C.T. Atlas Computer', Proc. IFIPS Congress, 1968.

B4 H. Beilner, 'Problems in Calibrating and Validating Simulation Models' Proceedings of 2nd Seminar on Experimental Simulation, Liblice (CSSR), 1973.

B5 H. Beilner, ' The Method of Good Balance', Proc. Computer Systems Performance Evaluation Workshop, Iowa State University, 1973.

B6 H. Beilner, 'Validating Simulation Models', Proc. Computer System Performance Measurement and Evaluation Workshop, Rocquencourt, 1974.

B7 H. Beilner and G. Waldbaum, 'Statistical Methodology for Calibrating a Trace-Driven Simulator of a Batch Computer System', page 423 in Reference F2.

B8 L.A. Belady, 'A Study of Replacement Algorithms for a Virtual Storage Computer, 'IBM Systems Journal, Vol.5 No.2, 1966.

B9 L.A. Belady and C.J. Kuehner, 'Dynamic Space Sharing in Computer Systems', Communications ACM, May 1969.

B10 C. Boksenbaum, S. Greenberg and C. Tillman, 'Simulation of CP-67', IBM Report No.G320-2083, 1973.

B11 A.J. Bonner, 'Using System Monitor Output to Improve Performance', IBM Systems Journal, Vol.8, No.4, 1969.

B12 P. Bookman, B. Brotman and K. Schmitt, 'Use Measurement Engineering for Better System Performance', Computer Decisions, April 1972.

B13 W. Buchholtz, 'A Synthetic Job for Measuring System Performance', IBM Systems Journal, Vol.8, No.4, 1969.

B14 J.P. Buzen and P.S. Goldberg, 'Guidelines for the use of Infinite Source Queuing Models in the Analysis of Computer System Performance', Proc. AFIPS National Computer Conference, 1974.

C1 P. Calingaert, 'System Performance Evaluation: Survey and Appraisal', Communications ACM, January 1967.

C2 P.H. Callaway, 'V.M. Monitor', IBM Research Report No. RC 4666, 1973.

C3 P.H. Callaway, 'VM/370 Performance Tools', IBM Systems Journal, Vol.14, No.2, 1975.

C4 P.S. Cheng, 'Trace-driven System Modelling', IBM Systems Journal, Vol.8, No. 4, 1969.

C5 E.G. Coffman and P.J. Denning, 'Operating Systems Theory', Prentice Hall, 1973.

C6 E.G. Coffman and T.A. Ryan, 'A Study of Storage Partitioning using Mathematical Model of Locality', Proc. Third ACM Symposium on Operating System Principles, 1971.

C7 Control Data 6400/6500/6600/6700 Computer Systems Reference Manual, Publication No. 60100000.

C8 Control Data 6000 Series Kronos Batch User's Manual, Publication No. 591506000.

C9 Control Data 6000 Series Kronos Timesharing User's Manual, Publication No. 59151300.

D1 C. Daniel and F. Wood, 'Fitting Equations to Data', Wiley, 1971.

D2 P.J. Denning, 'The Working Set Model for Program Behaviour', Communications ACM, May 1968.

D3 M.A. Diethelm, 'A Method of Evaluating Mass Storage Effects on System Performance', Proc. National Computer Conference, 1973.

D4 C.E. Dingley, 'Performance Measurement and Analysis of an Interactive System under a Controlled Workload', M.Sc. Thesis, Imperial College, London,1973.

D5 N. Draper and H. Smith, 'Applied Regression Analysis', Wiley, 1966.

D6 M.E. Drummond, 'Evaluation and Measurement Techniques For Digital Computer Systems', Prentice-Hall, 1973.

F1 G.S. Fishman, 'Concepts and Methods of Discrete Event Digital Simulation', Wiley, 1973.

F2 W. Freiberger (ed.), 'Statistical Computer Performance Evaluation', Academic Press, 1972.

F3 H.P. Friedman and G. Waldbaum, 'Evaluating System Changes under Controlled Workload Conditions: A Case Study', Proc. Computer Systems Performance Evaluation Workshop, Iowa State University, 1973.

G1 E. Gelenbe, 'On Approximate Computer System Models, Journal ACM, January 1975.

G2 R. Gimbel and H. Schwetman, 'Measurement Techniques for Computer System Performance Evaluation', Proc. Computer Systems Performance Evaluation Workshop, Iowa State University, 1973.

G3 H. Gomaa, 'Computer System Evaluation for Selection Purposes', Imperial College,Dept. of Computing and Control, Report No. 73/14, May 1973.

G4 H. Gomaa, 'Performance Measurement and Analysis of CDC 6000 Systems', Imperial College, Dept. of Computing and Control, Report No. 74/28, March 1974.

G5 H. Gomaa, 'An Exercise in Resource Allocation', Software - Practice and Experience, Vol. 4, No. 3, 1974.

G6 H. Gomaa and M.M. Lehman, 'Performance Analysis of an Interactive Computing System in a Controlled Environment', Proc. Online Conference on Computer System Evaluation, September 1973.

G7 U. Grenander and R. Tsao, 'Quantative Methods for Evaluating Computer System Performance: A Review and Proposals', Page 3 in F2.

H1 G. Holtwick, 'Designing a Commercial Performance Measurement System', Proc. ACM/SIGOPS workshop on System Performance Evaluation, April, 1971.

H2 P.H. Hughes and F. Moe, 'A Structural Approach to Computer Performance Analysis', Proc. AFIPS National Computer Conference, 1973.

I1 IBM System/360, Scientific Subroutine Package, Publication No. H20-0205.

I2 IBM System/370, OS/VS Service Aids, Chapter 1 - Generalised Trace Facility, Publication No. GC28-0633.

I3 IBM System/370, OS/VS System Management Facility, Publication No. GC35-0004.

J1 P.G. Jones, 'Performance Analysis of the Batch and Interactive Environments of a Multiprogramming System', M.Sc. Thesis, Imperial College, London, 1974.

J2 E.O. Joslin and J.J. Aiken, 'The Validity of Basing Computer Selection on Benchmark Results', Computers and Automation, January 1966.

K1 B. Kernighan and P. Hamilton, 'Synthetically Generated Performance Test Loads for Operating Systems', Proc. 1st SIGME Symposium on Measurement and Evaluation, 1973.

K2 S.R. Kimbleton, 'The Role of Computer System Models in Performance Evaluation', Communications ACM, July 1972.

K3 S.R. Kimbleton, 'A Fast Approach to Computer System Performance Prediction', Proc. Computer Architectures and Networks - Modelling and Evaluation Workshop, IRIA, 1974.

K4 L. Kleinrock, 'A Continuum of Time Sharing Scheduling Algorithms', Proc. AFIPS Spring Joint Computer Conference, 1970.

L1 K. Landau, 'Acquisition and Analysis of Raw Accounting Data', Proc. EuroComp Congress, May 1974.

L2 M.M. Lehman and H. Gomaa, 'Interactive System Performance in a Simulated Environment', Imperial College, Dept. of Computing and Control, Report No. 73/9, May 1973.

L3 M.M. Lehman and J.L. Rosenfeld, 'Performance of a Simulated Multiprogramming System, 'Proceedings AFIPS Fall Joint Computer Conference, 1968.

L4 P.J. Letts, 'Characteristics of the Central Computing Workload at CERN on a CDC 6600', CERN Data Handling Division, Report No. DD/71/22, 1971.

L5 P.J. Letts, 'Peripheral Activity on CDC 6000 Series Machines Used at CERN', CERN Data Handling Division, Report No. DD/72/9, 1972.

L6 H.C. Lucas, 'Performance Evaluation and Monitoring', ACM Computing Surveys, Vol.3, No.3, 1971.

M1 J.M. McKinney, 'A Survey of Analytical Time-Sharing Models', ACM Computing Surveys, Vol. 1, No. 2, 1969.

M2 M.F. Morris, 'Kiviat Graphs - Conventions and Figures of Merit', ACM Performance Evaluation Review, October 1974.

M3 J.E. Morrison, 'User Program Performance in Virtual Storage Systems', IBM Systems Journal, Vol. 12, No. 3, 1973.

N1 N.R. Nielson, 'Computer Simulation of Computer System Performance', Proc. 22nd ACM National Conference, 1967.

N2 J.D. Noe, 'Acquiring and Using a Hardware Monitor', Datamation, April 1974.

N3 J.D. Noe and G.J. Nutt, 'Validation of a Trace-driven CDC 6400 Simulation', Proc. AFIPS Spring Joint Computer Conference, 1972.

O1 P. Oliver, G. Baird, M. Cook, A. Johnson and P. Hoyt, 'An Experiment in the use of Synthetic Programs for System Benchmarking', Proc. AFIPS National Computer Conference, 1974.

P1 C.C. Parish, 'On the Application of Queuing Theory to Analysing On-line Computing Systems', The Computer Journal, May 1975.

P2 T. Pinkerton, 'Performance Monitoring in a Time-sharing System', Communications ACM, November 1969.

R1 J. Rodriguez-Rosell and J.P. Dupuy, 'The Evaluation of a Time-sharing Page Demand System', Proc. AFIPS Spring Joint Computer Conference, 1972.

S1 J. Saltzer and J. Gintell, 'The Instrumentation of Multics', Communications ACM, August 1970.

S2 M. Schatzoff and P. Bryant, 'Regression Methods in Performance Evaluation: Some Comments on the State of the Art', Proc. Computer System Performance Evaluation Workshop, Iowa State University, 1973.

S3 J.M. Schwartz and S.M. Wyner, 'Use of the SPASM Software Monitor to Evaluate the Performance of the Burroughs 6700', Proc. AFIPS National Computer Conference, 1973.

S4 H.D. Schwetman, 'A Study of Resource Utilisation and Performance Evaluation of Large-scale Computer Systems', Ph.D. Thesis, University of Texas, Austin, July 1970.

S5 P.H. Seaman and R.C. Soucy, 'Simulating Operating Systems', IBM Systems Journal, Vol. 8, No. 4, 1969.

S6 P.R. Sebastian, 'HEMI-Hybrid Events Monitoring Instrument', ACM Performance Evaluation Review, December 1974.

S7 S. Sherman, F. Baskett and J.C. Browne, 'Trace-driven Modelling and Analysis of CPU Scheduling in a Multiprogramming System', Communications ACM, December 1972.

S8 S. Siegel, 'Non-Parametric Statistics for the Behavioural Sciences', McGraw-Hill, 1956.

S9 G.W. Snedecor and W.G. Cochran, 'Statistical Methods', Iowa State University Press, 1967.

S10 R. Snyder, 'A Quantitative Study of the Addition of Extended Core Storage', ACM Performance Evaluation Review, March 1974.

S11 K. Sreenivasan and A.J. Knielman, 'On the Construction of a Representative Synthetic Workload', Communications ACM, March 1974.

S12 W.I. Stanley, 'Measurement of System Operational Statistics', IBM Systems Journal, Vol.8, No.4, 1969.

S13 D. Stevens, 'System Evaluation on the CDC 6600', Proc. IFIPS Congress, 1968.

T1 T.J. Teorey and T.B. Pinkerton, 'A Comparative Analysis of Disc Scheduling Algorithms', Proc. Third Symposium on Operating Systems, 1971.

T2 J.L. Thompson, 'Workload Characteristics of a University Computing Environment, M.Sc. Thesis, Imperial College, London, 1973.

T3 E.M. Timmreck, 'Computer Selection Methodology', ACM Computing Surveys, Vol. 5, No. 4, 1973.

W1 G. Waldbaum, 'Evaluating Computing System Changes by Means of Regression Models', Proc. 1st SIGME Symposium on Measurement and Evaluation, 1973.

W2 G. Waldbaum and H. Beilner, 'SOUL - A Simulation of OS Under LASP', Proc. Summer Simulation Conference, 1972.

W3 G. Waldbaum and H. Beilner, 'Submodel Simulation', Proc. Summer Simulation Conference, 1973.

W4 R. Watson, 'Computer Performance Analysis: Applications of Accounting Data', Rand Report R-573-NASA/PR, May 1971.

W5 P. Whitehead, 'Job Management under Kronos 2.1', Proc. European Control Data Users Conference, September 1974.

W6 J. Winograd, S.J. Morganstein and R. Herman, 'Simulation Studies of a Virtual Memory, Time-shared, Demand Paging Operating System', Proc. Third ACM Symposium on Operating System Principles, 1971.

W7 K. Wong and J.C. Strauss, 'Use of a Software Monitor in the Validation of an Analytical Computer System Model', Software-Practice and Experience, Vol. 4, No. 3, 1974.

Z1 F.W. Zurcher and B. Randell, 'Iterative Multilevel Modelling - A Methodology for Computer System Design', Proc. IFIPS Congress, 1968.

APPENDIX A: INTRODUCTION TO MULTIPLE LINEAR REGRESSION ANALYSIS

A.1 Introduction

This appendix is intended for readers who are unfamiliar with regression analysis or for those who wish to refresh themselves on the subject. The treatment here is meant to be of sufficient detail to enable unfamiliar readers to follow chapters 6 and 7.

An overview of multiple linear regression analysis is presented in section A.2. The procedures for selecting the 'best' regression equation are reviewed in section A.3, and the procedure implemented in this project is discussed. The implementation aspects of the regression analysis are described in section A.4.

A.2 Multiple Linear Regression Analysis

A.2.1 Regression Analysis

Regression analysis is an empirical method for analysing workload and performance data. Like many other empirical methods, it is a statistical method of analysing data.

A regression model deals with the following problem. Given a set of data containing the observations for several input (independent) variables X_1, X_2 ... X_k and an output (dependent) variable, it is required to fit the data by means of the function:

$$Y = f(X_1, X_2 \ldots X_k)$$

The functional relationship could be realised by means of a linear model:

$$Y = a_0 + \sum_{i=1}^{k} a_i X_i \qquad (1)$$

where a_i, i = 0, 1, ... k are unknown parameters. These parameters are called the regression coefficients and may be determined by least square fitting techniques.

A.2.2 Method of Least Squares

The method of Least Squares states: Find the values of the constants in the chosen equation that minimise the sum of the squared deviations of the observed values from those predicted by the equation (D1). An added requirement for linear least squares estimation is that the equations chosen be linear in the regression coefficients, a_o, a_1 ... a_k, hence the term linear regression (D1).

Linear least squares estimation partitions the total variability in the data (expressed as the total sum of squares) into two parts: the regression sum of squares (i.e. due to the fitted equation) and the residual sum of squares.

Many different forms of models may be built using linear least squares estimation. A simple linear regression model has one independent variable. A multiple linear regression model has two or more independent variables. A multiple linear regression model may also have second order terms, e.g.

$$Y = b_0 + b_1X_1 + b_{11}X_1^2 + b_2X_2 + b_{22}X_2^2 + b_{12}X_1X_2$$

A.2.3 Assumptions of the Least Squares Method

Certain assumptions are made in the least squares method of regression analysis. A very important assumption is that the data used for constructing the model is typical, that is the data is a representative sample

from the whole range of situations it is required to analyse and model. It should be obvious that an equation which represents the typical behaviour of a system cannot be derived from non-typical data (D1).

A number of the assumptions made concern the properties of the residuals (or errors) in the model. The residual e_i in the model is the difference between the observed value of the dependent variable y_i and the predicted value Y_i,

$$e_i = y_i - Y_i$$

The assumptions are that the errors are independent and that their distribution is normal with zero mean and fixed variance.

Another assumption is that the values of the independent variables are known without error. All the error is in the values of the dependent variable.

According to Daniel and Wood (D1), a further unwritten assumption is "that the data used are 'good' data. But most large collections of data and occasionally even small collections contain a few 'wild points' called mavericks or outliers. What happens to make them nontypical cannot usually be reconstructed. They must be spotted however, since to retain them may invalidate the judgements we make."

Consequently, it is important to carry out an analysis of the data used in constructing the model and of the residuals obtained from the constructed model, to determine if any outliers exist.

A.2.4 Measures of Goodness of Fit

Once the regression model has been built, it is necessary to determine how well the chosen equation fits the data. A number of complementary measures are available:

(i) The multiple correlation coefficient squared, R^2, represents the fraction of the total variation explained by the fitted equation. R^2 is the ratio of the regression sum of squares to the total sum of squares.

(ii) The F-test is used to judge the statistical significance of the value of R^2.

(iii) The t-test for the regression coefficient of each independent variable in the model is used to judge whether the independent variable makes a significant contribution to the model.

The validity of the F and t tests depends on the assumption that the residuals are normally distributed. An analysis of residuals should be carried out to test for normality and to determine if there are any outliers in the data.

A.3 Selecting the 'Best' Regression Equation

A.3.1 Introduction

When constructing a regression model, there is one dependent variable Y and a number of independent variables $X_1 \ldots X_m$. It is possible to construct a model containing all m independent variables. However it is likely that some of the independent variables make no significant contribution to the model, in which case there seems little point in including them in the model . Furthermore, their presence in the model is misleading, since it may suggest that these variables are making a useful contribution.

Consequently, a number of statistical procedures have been devised for selecting only a subset of the independent variables to include in the model. There is no unique statistical procedure for doing this and some personal judgement is necessary. Furthermore, the different methods do not all lead to the same solution when applied to the same problem (D5).

A.3.2 The Forward Selection Procedure

One of the procedures used for selecting the 'best' equation is the Forward Selection procedure (D5) in which independent variables are introduced one at a time into the equation. This is also referred to as the stepwise regression procedure in some texts (D1). The steps involved in the forward selection procedure are as follows:

(1) The first variable introduced is the independent variable most highly correlated with the dependent variable Y.

(2) The partial correlation coefficients between the remaining independent variables and Y are then found. The independent variable with the highest partial correlation coefficient with Y is selected next.

(3) Each time a variable is introduced into the model, R^2 is determined. Some criterion is then used to determine whether the addition of the new variable has made a significant improvement to the model. If the improvement is significant, steps 2 and 3 are repeated.

(4) If the improvement is not significant, then the selection procedure is stopped. The criterion for judgement could be a partial F-test for the

variable most recently entered. Alternatively, the criterion could be whether R^2, the fraction of the variation explained by the model, had been improved by a specified amount, e.g. 1%.

A.3.3 The Stepwise Regression Method

The Stepwise Regression procedure is an improved version of the forward selection procedure (D5). In the Forward Selection procedure, it is possible that the introduction of a variable into the model results in the contribution made by a variable already in the model to no longer be significant. This is particularly likely if some of the independent variables are correlated.

Because of this, the Stepwise Regression procedure does one further test at each step. Each time a variable is introduced into the model, a partial F-test is made on those variables already in the equation to determine whether they still make a significant contribution. If a variable no longer does, then it is eliminated from the equation.

A.3.4 The Procedure Implemented

The method used for this project was the Forward Selection procedure. This was dictated by its presence in the main statistical program library on the Kronos system, namely the IBM Scientific Subroutine Package (I1). However, by exercising some personal judgement, the same results may be obtained using this procedure as with the Stepwise Regression procedure. After a run using the Forward Selection procedure, the model selected is inspected. The t-tests on each of the regression coefficients of the independent variables indicate whether those independent variables make a significant contribution to

the model. Those variables that do not may be eliminated and a further run is carried out. This process may be repeated until all the independent variables make a significant contribution to the model. This model is then identical to the model which would have been selected using the Stepwise Regression procedure.