The design of calibration experiments for synthetic jobs

W. T. Graybeal* and U. W. Pooch

Industrial Engineering Department, Texas A & M University, College Station, Texas 77843, USA

The workload processed by a computer system can have a dramatic impact on the system's measured performance. A computer performance evaluation study conducted using empirical techniques requires an executable test workload. A test workload composed of synthetic jobs appears the most promising. Such a synthetic mix requires development of predictor equations which allow the setting of the parameters to produce the desired resource demand patterns.

A statistical methodology composed of experimental design techniques and regression analysis is proposed to aid in the development of prediction equations relating the resource descriptor variables and the synthetic job parameters. Two synthetic jobs are posed and the experimental design procedure is demonstrated with workload data from the Amdahl 470/V6 at Texas A&M University. Some factors related to validation of test workloads are presented.

(Received July 1979; revised February 1980)

1. Introduction

Computer Performance Measurement and Evaluation (CPME) studies have been classified by Lucas (1971) according to the reasons for which they are conducted. These reasons include selection of new systems (*selection* studies), projection of the effect of proposed hardware/software modifications (*projection* studies) and improvement of the level of performance of a current system (*performance monitoring*). Such evaluation studies can be further characterised according to the evaluation techniques employed. Three general techniques have emerged for the evaluation of computer systems: analytical, simulation, and empirical (Grenander and Tsao, 1972).

Central to any evaluation study is the test workload which is being processed by the system under consideration. If analytical techniques are employed, the test workload will normally be represented in the form of interarrival and service distributions (Coffman and Denning, 1972). A simulation study requires an abbreviated job description in a form compatible with the simulator. An empirical study, on the other hand, requires an executable workload.

A number of different types of executable test workloads have been proposed (Ferrari, 1978). These include *benchmarks*, *instruction mixes*, *standard jobs* and *synthetic jobs*. Synthetic jobs offer advantages in the area of flexibility and portability over instruction mixes and standard jobs. They also avoid the security and privacy problems associated with using real jobs (benchmarks). A test workload composed of synthetic jobs, then, is likely to be the most useful form of an executable test workload.

One of the primary criteria applied in assessing the usefulness of a test workload is how accurately it reflects the resource demands of the real workload which spawned it. A test workload which accurately reflects the characteristics of the real workload is said to be *representative*. Constructing a representative test workload using synthetic jobs requires careful design of the jobs making up the mix. Some of the techniques and procedures useful in designing synthetic jobs will be surveyed in this paper. Most of the techniques surveyed are oriented toward test workloads constructed for a batch processing installation. Similar considerations apply to transactions in a time sharing environment; however the general form of the model is different. The actions which must be emulated in an interactive session include user log-on, program creation, editing, program compilation, program execution and user log-off. A model embodying such actions can more realistically be referred to as an interactive script (Ferrari, 1978) than a synthetic job.

Although considerable work has been done with synthetic jobs (Buchholz, 1969; Sreenivasan and Kleinman, 1974), much remains to be done. One major problem is determining the settings of the various parameters to produce a desired resource demand pattern. The primary purpose of this paper is to propose a systematic design of experiments to calibrate synthetic jobs. Existing statistical experimental design techniques appear applicable in this regard.

2. General considerations in the design of synthetic jobs

A synthetic job is a parametric program in which the demands placed upon the various system resources are controlled by the values assumed by various input variables (parameters) (Ferrari, 1978). This relationship to the actual resource utilisation requires the programmer to approach the design of synthetic jobs from a different viewpoint to normal programming problems. Normal programming projects are usually undertaken for a particular reason. That is, the user wants the computer to perform a particular task. The task to be performed is the over-riding consideration in program development. There may be an attempt to minimise the resources used in an effort to hold down the cost of the project, but this is generally a secondary consideration. Synthetic jobs, on the other hand, are independent of the task which is performed. They are also independent of any input data or data files accessed by the real programs they are designed to emulate. The sole consideration in their design is that they use the same amount and types of resources that their real counterparts use. Thus, a somewhat arbitrary 'compute loop' can be used to force the synthetic job to consume a particular amount of CPU time. I/O activity by real jobs can be emulated by having the synthetic job access arbitrary files of the required type (e.g. tape, disc or card). These files can be 'garbage files' expressly constructed for this purpose, or any other file to which the analyst has access. Thus, there is no unique synthetic job for each situation. A multitude of logically different programs can be forced to exhibit the same resource demand patterns with the proper choice and setting of parameters.

The degree of complexity of a synthetic job is generally determined by the level of detail used in characterising the real workload. If a limited resource descriptor set is used, a relatively simple synthetic job will normally suffice. If, on the other hand, an expanded resource descriptor set is used which reflects more minute aspects of the real job's resource utilisation, a more complex synthetic job will generally be required. Ferrari (1978)

*US Air Force Academy.

CCC-0010-4620/81/0024-0132 $03.50

illustrated this point with two examples: this first example concerns construction of a test workload for a batch processing installation. Jobs in the workload were characterised by the descriptor pair (t_{cpu}, n_{io}). The first descriptor gives the CPU time required by the job while the second gives the number of I/O operations initiated by the job. Since the type of I/O is not specified, it can be assumed to be simple 'reads' from cards and 'writes' to a printer (or any other mode for that matter) in an arbitrary proportion. This is not to say that it is not important or ncecessary to distinguish between various types of I/O. Rather, it indicates that various types of I/O and the relative proportion of each type performed by a job are not distinguishable using this limited description set. An expanded descriptor set would normally be necessary to reflect the I/O activity of a given job accurately. A synthetic job designed to emulate jobs for which such a limited descriptor set is adequate (if any exist) can be composed of a simple loop. I/O is performed a certain proportion of the iterations through the loop, and some arbitrary computation performed some other (or perhaps the same) proportion of the times through the loop. The loop is executed until the required number of I/O operations are performed and the proper amount of CPU times is accrued. An example of such a synthetic job and a situation in which this low level of detail is sufficient is given in the case study.

More complex synthetic jobs are typefied by the one developed and tested by Buchholz (1969). This job is designed to emulate a file processing action. There are three parameters used, which specify the number of master records read in, the number of detail (transaction) records processed and the number of times the 'compute' loop is executed. This job can be used to emulate the resource demands of jobs whose resource descriptor set is somewhat expanded over the earlier one described. An example of the use of such a synthetic job is also given in the case study.

2.1 *Parameters of synthetic jobs*

The parameters of a synthetic job allows the individual system resource demands to be easily modified. In general, greater flexibility requires more parameters, while simplicity and economy dictate that the number of such parameters be kept to a minimum. In the final analysis, it is the level of detail used in characterising the real workload which determines the number of parameters to use. This required level of detail is in turn determined by the resolution necessary in the evaluation study. For example, consider a test workload composed of synthetic jobs where each synthetic job has parameters to specify memory size and total CPU processing time. This workload might be sufficient if the aim of the evaluation study is to determine the effects of altering main memory on CPU utilisation. It would not provide the required resolution if the aim of the study is to determine the effects of differing amounts of I/O processing on CPU and I/O overlap. In fact this latter study would require at least one parameter to allow the ratio of CPU processing to I/O processing to be altered. It may also be necessary to include resource descriptor variables which specify the duration and relative timing of I/O requests. Thus, there is a three-way dependence among the performance measures observed in the study, the descriptor variables used to characterise jobs in the workload and the synthetic job parameters used to control the demands placed on various system resources.

More formally, suppose that a test workload W_t is constructed for use in an evaluation study in which the l performance variables $V_1, V_2, \ldots, V_l$ are to be observed. Suppose further that these performance variables are functions of m system resources described by the descriptor variables $r_1, r_2, \ldots, r_m$, and that the values assumed by these descriptor variables are determined by n user parameters $p_1, p_2, \ldots, p_n$. The relations existing among the variables can be expressed as

$$\begin{aligned}
V_1 = V_1(r_1, \ldots, r_m) &= V_1[r_1(p_1, \ldots, p_n), \ldots, r_m(p_1, \ldots, p_n)] \\
&= V_1(p_1, \ldots, p_n) \\
V_2 = V_2(r_1, \ldots, r_m) &= V_2[r_1(p_1, \ldots, p_n), \ldots, r_m(p_1, \ldots, p_n)] \\
\vdots \qquad\qquad &= V_2(p_1, \ldots, p_n) \qquad \vdots \\
V_l = V_l(r_1, \ldots, r_m) &= V_l[r_1(p_1, \ldots, p_n), \ldots, r_m(p_1, \ldots, p_n)] \\
&= V_l(p_1, \ldots, p_n).
\end{aligned}$$

The relations can be summarised in more compact vector notation as $\vec{V} = \vec{V}(\vec{r}) = \vec{V}[\vec{r}(\vec{p})] = \vec{V}(\vec{p})$. Now, recognising that the values assumed by the parameters $p_1, \ldots, p_n$ completely determine W_t, the composite relation $V_i = V_i(W_t)$, $i = 1, \ldots, l$, (or $\vec{V} = \vec{V}(W_t)$) results, where $W_t = W_t(p_1, \ldots, p_n)$.

One problem which must be solved in constructing W_t is determining the relationship which exists between the resource descriptor variables $r_1, \ldots, r_m$ and the synthetic job parameters $p_1, \ldots, p_n$. The parameters $p_1, \ldots, p_n$ can be assumed to be independent of one another, and in some cases they may bear a simple linear relationship to the r_i's. This relationship can be established by observing the r_i's for a few runs of the synthetic job with varying p_i's, and applying regression analysis (Draper and Smith, 1966). The linear form of the relationships $r_i = r_i(p_1, \ldots, p_n)$, $i = 1, 2, \ldots, m$, allows inversion to give relationships of the form $p_j = p_j(r_1, \ldots, r_m)$, $j = 1, \ldots, n$. This assumes $n \geqslant m$ and that the original system is nonsingular. These latter relations can be used to determine the appropriate parameter settings to produce a given resource demand pattern.

Examples of the use of linear regression in establishing the relationships which exist between the resource descriptor variables $r_1, r_2, \ldots, r_m$ and the synthetic job parameters $p_1, p_2, \ldots, p_n$ are given in the case study. It should be noted that the simple form of these relations does *not* suggest that similar simple relationships exist between the performance variables $V_1, V_2, \ldots, V_l$ and the resource descriptor variables $r_1, r_2, \ldots, r_m$. Establishing this relationship must be accomplished during the evaluation study itself.

2.2 *Controlling the demand for system resources*

A procedure for establishing the relationship between the resource descriptor variables $r_1, r_2, \ldots, r_m$ and the synthetic job parameters $p_1, p_2, \ldots, p_n$ was suggested in the previous section. This procedure assumes that parameters which are likely to affect the job's demand for a given resource have been established and incorporated into the design of the synthetic job. Some of the ways in which the demands placed upon system resources can be controlled are surveyed in this section. It should be noted that some of these techniques are system dependent and their use severely limits the usefulness of the program in other than performance monitoring studies.

One of the major system resources is main memory. The amount of main memory used by a given job is obviously related to the size of the program as well as the space needed for system routines supporting the job's execution. A job's main memory requirements can thus be altered by modifying the size of arrays or by including routines which may never be called. A number of systems (i.e. IBM) enforce a policy known as 'preallocation of resources' to preclude deadlock problems (Coffman and Denning, 1973). The maximum amount of main memory likely to be used by the job must be requested in advance of its initiation. If this requested amount is not sufficient to allow program execution, the job is terminated. The size of the region in main memory allocated to a particular program, if such a strategy is employed, can be either increased or decreased by altering the region request field in the job control statements.

Control of the amount of CPU processing time used by a program is possible by including a 'compute-loop' control parameter. An arbitrary sequence of computations is performed iteratively until the desired CPU time is accrued. The required number of iterations through the loop can be controlled precisely through access to system timers (Ferrari, 1978). It can alternately be established in advance through calibration experiments. The amount of processing time accrued by a particular job is related to factors other than simply the number of computations performed. The number of I/O activities initiated, for example, can have a significant impact on CPU time used.

Control of the I/O processing requirements of a job is more complicated than the control of either main memory or CPU time because of the multitude of different types of I/O. It may be necessary to control each of them, depending upon the resolution needed in the study. Unit record I/O (i.e. cards read, lines printed and cards punched) is the easiest to control. The number of cards read is obviously a direct function of the size of the program. It can be varied, within certain limits, by including or excluding comment and data cards. The number of lines printed (or cards punched) can be controlled through inclusion of a 'print' (or 'punch') loop. This loop is executed a sufficient number of times to produce the desired output. Tape and disc (or drum) I/O is controlled by creating files which are accessed using the proper mode. Records can be read, modified and written under the control of a file processing loop. There is a potential problem in accurately reflecting the real workload's processing behaviour. This results from the fact that in addition to controlling the number of I/O activities initiated, the size of the data block transferred each time must also be specified. Data on the real workload's resource demands is generally not available at the required level of detail from system accounting logs. It can be obtained by using a monitor.

Another type of I/O activity which must be controlled in virtual memory systems is paging I/O. In a demand paging environment, blocks of data are transferred from auxiliary storage into main memory as required. If main memory is full, some 'pages' may have to be recopied back to auxiliary storage to make room for the next 'page' copied into main memory. Paging I/O is not an activity which can be controlled directly in the same fashion as other resource demands. Paging activity of a given job is dependent not only on the internal structure of the program, but also on the environment which exists at the time. Some work has been done on influencing paging activity in the development stage of a program. Techniques useful in improving the locality of a program and thus decreasing its expected page fault rate are discussed by Spirn (1977). However, due to the environmental dependence, any significant control over paging activity will probably have to be exerted during the calibration/validation phase when the entire test workload is available.

Direct control can be exerted over many of the system resources through inclusion of loop control parameters and proper job control statements. An example of the use of parameters to control the various system resources is included in the case study.

3. The design of calibration experiments

It is necessary, once a synthetic job has been designed, to establish the relationship between the parameters of the synthetic job and the resource descriptor variables used to characterise jobs in the real workload. Such a process can be termed 'calibrating' the synthetic job. The procedure proposed in an earlier section requires that the synthetic job is executed on the system for various parameter settings. The corresponding values of the descriptor variables are recorded for each run, and regression analysis is used to establish the desired relationship. There are a number of unanswered questions associated with this procedure. These include 'how many runs of the synthetic program are necessary to establish an accurate relationship?', 'what parameter settings should be used for each run?', and 'how to account for the acknowledged environmental variations in the resource demands from one run to the next'. The use of statistical experimental design techniques is proposed in this section to assist in answering these questions.

The magnitude of the demands placed on system resources by a given job can vary from one run to the next. Some of the demands most susceptible to these environmental differences are CPU processing time, I/O processing time and data transfer over the channels handling paging activity. This variation in resource demands can have a significant effect on relationships established through regression analysis. Indiscriminate running of the synthetic job will yield data in which it is impossible to separate the effect on the response variable due to this 'chance' variation from that caused by the setting of various parameter levels.

Most of the parameters used in controlling the magnitude of the demands placed upon various system resources by a synthetic job can assume a wide range of values. For example, the number of times a 'compute' loop is executed is constrained only to be a non-negative integer representable in the counter register of the machine. Similar restrictions (or lack thereof) apply to other parameters. Failure to use a wide enough range of values for these parameters will yield a predictor equation which cannot be used in some cases. This is because it is almost never feasible to extrapolate using a regression equation (Draper and Smith, 1966).

Related to the setting of the parameter levels for each run of the synthetic job is the required number of runs. The synthetic job could be run a large number of times (say 100) with the parameters set at the same values. This obviously would yield a highly reliable relationship for that particular combination of settings. The validity of the relationship for some other combination of parameter settings would be highly suspect, however.

Problems similar to those outlined above are commonly encountered in other data analysis situations. A branch of statistics known as experimental design (Hicks, 1973) has evolved to aid in the resolution of these problems. The methodology outlined for designing factorial experiments appears applicable to this problem.

A factorial experiment is one in which all levels of a given factor are combined with all levels of every other factor of the experiment (Hicks, 1973). Each of the synthetic job parameters to be varied can be considered as a factor in the calibration experiment. Levels for each factor can be established which are likely to cover the required range of resource demands. Each unique combination of factor levels can be thought of as a 'treatment' to be applied. Treatments are assigned at random to each run of the job.

The use of statistical design techniques provides a number of advantages in calibration experiments. They include:

(*a*) the randomisation of the treatment to run assignment minimises the effect of chance environmental variations in resource demands

(*b*) for a given number of factors and levels per factor, one can precisely calculate the number of runs necessary for a complete replication of the experiment. For example, if five factors are present, and each can assume two levels, $2^5 = 32$ runs are required. The analyst can reduce the number of runs by using fractional replications. This involves confounding some effects

(*c*) the significance of the effects on the resource demands by the various parameters can be tested through an analysis of

variance. Interaction effects can also be tested, although in some cases it is difficult to interpret such effects

(*d*) confidence limits can be established for the obtained regression coefficients.

It costs no more in most cases to conduct a carefully designed experiment than it does a poorly designed one. The use of statistical experimental design techniques can have a significant impact in the calibration phase.

3.1 *Validating the test workload*

The calibration experiments discussed in the previous section can be used to establish predictor equations relating the synthetic job parameters to the resource descriptor variables. A synthetic job mix can then be constructed by including sufficient copies of each of the synthetic jobs with the appropriate parameter settings. It is necessary to execute this synthetic mix on the system being studied and to determine what degree of representativeness has been achieved. This process can be termed validation.

A number of authors (Agrawala, Mohr and Bryant, 1976; Ferrari, 1978; Johnson, 1977; Screenivasan and Kleinman, 1974) have emphasised the importance of validating test workloads. The general consensus seems to be that a test workload which has not been validated should not be used. The particular subset of the real workload which is used as a model in the design of a test workload is selected because it exhibits some characteristics pertinent to the evaluation study (i.e. heavy loading, high paging rate, etc.). If the test workload does not exhibit the same characteristics, the evaluation study can be severely hampered.

If the test workload does not accurately reflect the resource demands of the real workload subset, it is probably due to

(*a*) errors in recording the resource demands, either because the recording process was not accurate or because the resource demand pattern was distorted (perhaps due to artifacts introduced by the monitoring process itself)

(*b*) errors introduced when the actual workload demands are reduced to probability distributions or clusters, or

(*c*) errors in computing the synthetic job parameters.

Errors of the first and second type are common to nearly all methods of generating test workloads. They can be precluded only by exercising extreme care in those stages of the construction process. Errors of the third type are unique to test workloads generated using synthetic jobs. Careful design of the calibration experiments should minimise the possibility of an error of this type occurring.

An obvious means of verifying the accuracy of the synthetic job parameters is to execute the test workload, record the demands placed upon the system resources, and then compare the resulting probability distributions of demand clusters with those produced by the real workload. A number of statistical tests (i.e. Chi-Square, Kolmogorov-Smirnov) are available for testing 'goodness of fit'. Errors of the first and second type mentioned above, however, could go undetected using this process. The monitoring process will be likely to introduce the same bias when the test workload is executed as it did during processing of the actual workload subset. The same analysis package is likely to be used to summarise both the resource demands of the actual workload and those of the test workload. Thus, the same errors are apt to occur in both analyses.

The validation phase of test workload construction is probably the least understood phase. There are a number of reasons for this. Many studies never progress this far, since it is the last phase of the process (although the calibration phase may be re-entered if a nonrepresentative test workload is produced). Secondly, to avoid distorting the demand characteristics of the workload, it must be executed in isolation from other jobs on the system. This requires a dedicated system during that period of time, which is sometimes inconvenient and expensive.

4. Case study

Statistical experimental design techniques appear to offer significant advantages in the design of calibration experiments. To illustrate the utility of these procedures, two separate synthetic jobs were designed. It should be noted that these jobs are used merely to illustrate the procedure. Thus some of the relationships between the parameter settings and resource descriptor variables may appear self-evident from the design of the programs. The first, a very simple job, was designed to emulate the resource demands of student jobs using the incore compilers (i.e. WATFIV, WATBOL, PLC). These jobs are hereafter referred to as Autobatch jobs. The second synthetic job was designed to emulate the resource demands of jobs using the standard OS translators. These jobs are referred to as Batch jobs.

The different types of jobs were analysed separately due to the severe restrictions placed upon the resource demands of Autobatch jobs. As an example, access to external files by such jobs is prohibited; a maximum of 6,000 lines of output can be produced. Thus, a limited resource descriptor set is adequate to distinguish between two such jobs. The descriptors used in this study included $X_1 \equiv$ number of cards read, $X_2 \equiv$ number of lines printed, and $X_3 \equiv$ CPU time used (0·01 sec).

The Batch jobs, on the other hand are not as restricted in the quantity and type of resources they can use. An expanded descriptor set is required for such jobs. Those descriptors used included $X_1 \equiv$ number of job steps executed, $X_2 \equiv$ total number of devices used by the job, $X_3 \equiv$ region size requested (kilobytes), $X_4 \equiv$ number of cards read, $X_5 \equiv$ number of lines printed, $X_8 \equiv$ number of cards punched, $X_7 \equiv$ number of pages read in, $X_0 \equiv$ number of pages read out, $X_9 =$ CPU time (0·01 sec), $X_{10} \equiv$ I/O time (0·01 sec), $X_{11} \equiv$ EXCP count to tape devices, and $X_{12} \equiv$ EXCP count to disc devices (excluding HASP pseudo devices). The EXCP count is the number of 'execute channel program' commands issued, and roughly corresponds to the number of blocks transferred.

Synthetic jobs designed to represent the Autobatch jobs can be very simple due to the limited resource descriptor set. The number of cards read is exactly determined by the number of source/comment statements in the program and the number of limits for a given synthetic job by either including or excluding data/comment cards. The number of lines printed can also be exactly controlled by including a print loop which is executed the desired number of times. The amount of CPU time is also related to the number of lines printed, hence this dependence must be accounted for. The synthetic job designed for the Autobatch jobs is described in Appendix 1.

The synthetic job designed for Autobatch has two parameters which may be varied to induce various resource demand patterns. These parameters are NRLIN ≡ the number of lines to be printed and NITER ≡ the number of times the compute loop is to be executed. The size of the program (number of cards read) was held constant throughout. These two parameters were used as 'treatments' in the experimental design used. Three 'levels' for each 'treatment' were established to cover the range of resource demands exhibited by the members of the Autobatch cluster. This results in nine unique treatment/level combinations. A completely randomised factorial design (3^2) was used to establish the parameter settings for the nine required runs of the job. The parameter settings for each run are shown in **Table 1.** The nine programs were run on the system and data collected which reflected the resource demands of each program. This data is summarised in **Table 2.**

The significance of the effect of varying NITER and NRLIN

Table 1 Parameter settings—Autobatch

	Run 1	2	3	4	5	6	7	8	9
NRLIN	50	50	50	150	150	0	0	150	0
NITER	50	5,000	2,500	50	2,500	5,000	50	5,000	2,500

Table 2 Resource demands—synthetic Autobatch job

	Run 1	2	3	4	5	6	7	8	9
X_1	33	33	33	33	33	33	33	33	33
X_2	88	88	88	188	188	38	39	188	38
X_3	5	77	41	9	43	74	3	80	38

Table 3 Parameter settings—Batch

	Run 1	2	3	4	5	6	7	8
NITER	1,000	0	1,000	0	0	0	1,000	1,000
NOUT	0	0	1,000	0	1,000	1,000	0	1,000
NTAP	0	1,000	1,000	0	0	1,000	1,000	0
NDIS	0	0	0	1,000	0	1,000	1,000	1,000

Table 4 Resource demands—synthetic Batch job

	Run 1	2	3	4	5	6	7	8
X_1	1	1	1	1	1	1	1	1
X_2	14	14	14	14	14	14	14	14
X_3	128	128	128	128	128	128	128	128
X_4	240	240	240	240	240	240	240	240
X_5	354	351	1,349	351	1,349	1,349	351	1,349
X_6	0	0	0	0	0	0	0	0
X_7	0	0	0	0	0	0	0	0
X_8	0	0	0	0	0	0	0	0
X_9	242	109	332	111	191	201	247	329
X_{10}	188	213	212	233	187	256	257	232
X_{11}	1	10	10	1	1	1	10	1
X_{12}	132	132	132	217	132	217	217	217

on X_3 was then tested. Both 'treatments' were found to be highly significant ($\alpha = 0{\cdot}0001$). The model used assumed no interaction between the parameters. The amount of CPU time used (X_3) was regressed on NITER and NRLIN, while the number of lines printed (X_2) was regressed on NRLIN. The following predictor equations were obtained through this regression:

$$X_2 = 38{\cdot}238 + 0{\cdot}998 \text{ NRLIN}$$

$$X_3 = 2{\cdot}399 + 0{\cdot}037 \text{ NRLIN} + 0{\cdot}014 \text{ NITER} .$$

The fit achieved by both regression equations was extremely good. The value of the multiple correlation coefficient (proportion of the variability accounted for) was 0·999978 for the equation relating X_2 to NRLIN, and 0·999679 for the equation relating X_3 to NRLIN and NITER.

Synthetic jobs designed to represent Batch jobs must be considerably more complex than those for Autobatch jobs due to the expanded resource descriptor set. A number of the 12 descriptor variables can be exactly controlled through Job Control Language (JCL) statements or the inclusion/exclusion of data/comment cards. Others must be controlled through parameters.

The synthetic job designed for Batch jobs (described in Appendix 1) has four parameters which can be varied to induce different resource demand patterns. They are NITER ≡ the number of times the compute loop is executed, NOUT ≡ the number of output lines produced, NTAP ≡ the number of records read from a tape file, and NDIS ≡ the number of records read from a disc file. Those resource demands which are not affected by varying these parameters were held constant throughout the experiment. In practice, these demands should be set to correspond with the real workload.

Two levels for each parameter were selected. A completely randomised factorial design (2^4) was used to establish the parameter settings for the various runs of the program. This design requires 16 runs to form one replication of the experiment. This was considered excessive due to the cost associated with each run. It was decided to use a fractional replication for this reason. A one half fractional replication requires only eight runs, but still allows testing of the main treatment effects. The effect of interaction among parameters was assumed negligible just as with the Autobatch experiment. Using the method illustrated by Hicks (1973), the 'treatment' combinations were divided into two blocks, with the four-way interaction effect confounded with the block effect. A coin flip was used to decide which of the blocks to use in the experiment. The parameter settings for the eight required runs of the job are listed in **Table 3.**

The synthetic jobs were run on the system and data collected reflecting the resource demands. This data is shown in **Table 4.**

The values for all 12 resource descriptors are shown; those which are not affected by the four parameters appear as constants. No attempt was made to control paging behaviour as this is largely environment dependent.

Table 4 shows that five of the 12 resource descriptors are affected by varying the four parameters. These are X_5 (number of lines printed), X_9 (CPU time used in 0·01 sec increments), X_{10} (I/O time used in 0·01 sec increments), X_{11} (EXCP count to tape devices), and X_{12} (EXCP count to disc devices). The significance of the effect of the parameters on the descriptor variables was tested. Using a level of significance $\alpha = 0{\cdot}05$, the effect on X_9 was significant for NITER and NOUT; the effect on X_{10} was significant for NOUT, NTAP and NDIS; the effect on X_5 was significant for NOUT; the effect on X_{11} was significant for NTAP; and the effect on X_{12} was significant for NDIS.

The descriptor variables were then regressed on those parameters which were identified as having a statistically significant effect. The resulting regression equations with the value of the multiple correlation coefficient indicated in parentheses are

$$X_5 = 351{\cdot}75 + 0{\cdot}99725\text{NOUT} \quad (R^2 = 0{\cdot}999997),$$

$$X_9 = 110{\cdot}00 + 0{\cdot}1345\text{NITER} + 0{\cdot}0860\text{NOUT} \quad (R^2 = 0{\cdot}998648),$$

$$X_{17} = 188{\cdot}25 - 0{\cdot}001\text{NOUT} + 0{\cdot}025\text{NTAP} + 0{\cdot}0445\text{NDIS} \quad (R^2 = 0{\cdot}999903),$$

$$X_{11} = 1{\cdot}00 + 0{\cdot}009\text{NTAP} \quad (R^2 = 1{\cdot}000000), \text{ and}$$

$$X_{12} = 132{\cdot}00 + 0{\cdot}085\text{NDIS} \quad (R^2 = 1{\cdot}000000).$$

The problem with inverting the above equations to yield predictor equations for the parameter setting is that there is one equation too many (i.e. five equations in four unknowns). The equation for X_{10} however is seen to be redundant, since I/O time is uniquely determined by the quantity and type of I/O

```
C     SYNTHETIC JOB TO SIMULATE AUTOBATCH
      ISEED=21151
      NRLIN=0
      NITER=50
      LIMIT=NRLIN
      IF(NITER.GT.LIMIT)LIMIT=NITER
      DO 10 I=1,LIMIT
      CALL URANDX(ISEED,IRAND,URAND)
      IF(I.GT.NRLIN)GO TO 20
      WRITE(6,600)URAND,I
600   FORMAT(1X,'THE RANDOM NUMBER IS',1X,F7.5,1X,'ON ITERATION NR',
     *1X,I5)
20    CALL URANDX(ISEED,IRAND,URAND)
      ISEED=IRAND
      URAND=URAND+COS(URAND)*SIN(URAND)
10    CONTINUE
      STOP
      END
      SUBROUTINE URANDX(JSEED,IRAND,URAND)
C     THIS ROUTINE GENERATES A PSEUDO-RANDOM STANDARD UNIFORM
C     VARIATE STARTING WITH INTEGER SEED JSEED, AND RETURNING
C     A NEW SEED IRAND AND STANDARD UNIFORM VARIATE URAND.
      IRAND=JSEED*65539
      IF(IRAND)5,6,6
5     IRAND=IRAND+2147483647+1
6     URAND=IRAND
      URAND=URAND*.4656613E-9
      RETURN
      END
```

Fig. 1

```
SYNTH:PROCEDURE OPTIONS (MAIN);
/*THIS SYNTHETIC PROGRAM IS DESIGNED
TO EMULATE THE RESOURCE DEMAND
CHARACTERISTICS OF BATCH JOBS WHICH
PERFORM TAPE, DISK, AND UNIT RECORD
I/O. INPUT PARAMETERS ARE
  NITER - THE NUMBER OF TIMES THE
          COMPUTE LOOP IS TO BE EXECUTED,
  NOUT - THE DESIRED NUMBER OF
          OUTPUT LINES,
  NTAP - THE DESIRED NUMBER OF
          TAPE EXCPS TO BE ISSUED,
  NDIS - THE DESIRED NUMBER OF
          DISK EXCPS TO BE ISSUED,
  JSEED - AN INTEGER SEED USED TO
          BEGIN THE FIRST RANDOM
          NUMBER STREAM,
  ISEED - AN INTEGER SEED USED TO
          BEGIN THE SECOND RANDOM
          NUMBER STREAM.  */
  ON FIXEDOVERFLOW ;
  ON ENDFILE (TAPIN) NTAP=0;
  ON ENDFILE (DISIN) NDIS=0;
/*DECLARE VARIABLES  */
  DECLARE  SYSIN    FILE STREAM INPUT,
           SYSPRINT  FILE STREAM OUTPUT,
           TAPIN    FILE RECORD INPUT,
           DISIN    FILE RECORD INPUT,
           COMMON   BIT(10000) VARYING,
           JSEED    FIXED DEC(9),
           ISEED    FIXED DEC(9),
           URAND    FLOAT DEC(6),
           IRAND    FIXED DEC(9),
           LIMIT    FIXED DEC(9),
           NITER    FIXED DEC(9),
           NOUT     FIXED DEC(9),
           NTAP     FIXED DEC(9),
           NDIS     FIXED DEC(9),
           (I,J,K)  FIXED BINARY(31),
           (A(5,5),B(5,5),C(5,5))    FLOAT DEC(6);
/* PREPARE FILES FOR PROCESSING  */
  OPEN FILE (TAPIN) INPUT;
  OPEN FILE (DISIN) INPUT;
  OPEN FILE (SYSIN) INPUT;
  OPEN FILE (SYSPRINT) OUTPUT;
/*  READ INPUT PARAMETERS  */
/*  COMPUTE CONTROL PARAMETERS  */
  NITER=0;
  NOUT=0;
  NTAP=0;
  NDIS=0;
  JSEED=51121;
  ISEED=21151;
  LIMIT=NITER;
  IF NOUT>LIMIT THEN LIMIT=NOUT;
  IF NTAP>LIMIT THEN LIMIT=NTAP;
  IF NDIS>LIMIT THEN LIMIT=NDIS;
/*  COMPUTE LOOP  */
MAINLOOP:DO I=1 TO LIMIT;
    CALL RAND(JSEED,IRAND,URAND);
    JSEED=IRAND;
/*  CHECK TO SEE IF OUTPUT LINE  */
    IF I < NOUT THEN
       PUT SKIP LIST ('RANDOM NUMBER IS',
          URAND,'ON ITERATION NR',I);
/*  CHECK TO SEE IF READ TAPE  */
    IF I < NTAP THEN
       READ FILE (TAPIN) INTO (COMMON);
/*  CHECK TO SEE IF READ DISK  */
    IF I < NDIS THEN
       READ FILE (DISIN) INTO (COMMON);
/*  CHECK TO SEE IF EXECUTE INNER LOOP  */
    IF NITER < I THEN GO TO PASS_IT;
/*  INNER COMPUTE LOOP  */
/*  THIS LOOP FILLS TWO 5X5 MATRICES
    A AND B WITH STANDARD UNIFORM
    RANDOM NUMBERS AND THEN INVOKES A
    ROUTINE TO MULTIPLY THE TWO MATRICES,
    RETURNING THE PRODUCT IN MATRIX C. */
INLOOP:DO J=1 TO 5;
          DO K=1 TO 5;
             CALL RAND(ISEED,IRAND,URAND);
             ISEED=IRAND;
             A(J,K)=URAND;
             CALL RAND(ISEED,IRAND,URAND);
             ISEED=IRAND;
             B(J,K)=URAND;
          END;
  END INLOOP;
PASS_IT:;
 END MAINLOOP;
MATMUL:PROCEDURE(A,B,C);
  DECLARE   (A(5,5),B(5,5),C(5,5))  FLOAT DEC(6);
  DECLARE   (X,Y,Z)  FIXED BINARY(31);
/*  THIS ROUTINE MULTIPLIES TWO 5X5
    MATRICES A AND B AND PRODUCES THE PRODUCT
    MATRIX C.  */
LOOP1:DO X=1 TO 5;
        DO Y=1 TO 5;
        C(X,Y)=0.0;
      END;
 END LOOP1;
LOOP2:DO X=1 TO 5;
        DO Y=1 TO 5;
          DO Z=1 TO 5;
            C(X,Z)=A(X,Y)*B(Y,Z)+C(X,Z);
          END;
        END;
 END LOOP2;
 RETURN;
END MATMUL;
RAND:PROCEDURE(JSEED,IRAND,URAND);
   DECLARE   (JSEED,IRAND)   FIXED DEC(9),
             URAND    FLOAT DEC(6);
   THIS ROUTINE GENERATES A PSEUDO-RANDOM
   STANDARD UNIFORM VARIATE URAND. THE STARTING
   SEED IS SUPPLIED THROUGH PARAMETER JSEED. A
   NEW SEED FOR THE NEXT PASS THROUGH THE
   PROCEDURE IS RETURNED THROUGH PARAMETER IRAND  */
 IRAND=JSEED*65539;
 IF IRAND < 0 THEN IRAND = IRAND+2147483647+1;
 URAND=IRAND;
 URAND=URAND*0.4656613E-9;
 RETURN;
END RAND;
END SYNTH;
```

Fig. 2

performed. Inverting the remaining relations yields the following predictor equations

$$\text{NOUT} = 1{\cdot}00276X_5 - 352{\cdot}72,$$
$$\text{NITER} = 7{\cdot}4349X_9 - 0{\cdot}6411X_5 - 592{\cdot}27,$$
$$\text{NTAP} = 111{\cdot}1111X_{11} - 111{\cdot}11, \text{ and}$$
$$\text{NDIS} = 11{\cdot}7647X_{12} - 1552{\cdot}95.$$

5. Summary

The use of statistical experimental design methodology in calibration experiments involving synthetic jobs can be quite beneficial. It costs no more in most cases to conduct a carefully designed experiment than it does a poorly designed one. The results obtained, however, are worth any added cost in time or effort.

Appendix 1

This appendix describes the two synthetic jobs designed as part of this study. The first job was developed to emulate the resource demands of Autobatch jobs. The resource descriptor set used to characterise the demands of Autobatch jobs contained only three elements, and hence the synthetic job is quite simple. The second job was designed to emulate the resource demands of Batch jobs. The expanded resource descriptor set used to characterise the Batch jobs necessitates a more complex synthetic job. Both programs contain features that make them somewhat system-dependent. For example, the random number generator depends on overflow in a 32-bit integer, while the Autobatch job is critically sensitive to the efficiency of implementation of the Sin and Cos Routines.

The synthetic job designed for Autobatch jobs is designed to allow the user to specify indirectly the number of lines printed and the total CPU time used by setting two parameters: NRLIN and NITER. The appropriate settings for these parameters may be determined using predictor equations established in an earlier section. A loop control parameter LIMIT = Maximum {NRLIN, NITER} is first calculated. The main loop is then executed a total of LIMIT times. The first NRLN times through the loop, an output line is produced. Other actions accomplished each time through the loop include calculating two pseudorandom numbers using a multiplicative congruential scheme and performing some simple calculations on the second of these two generated numbers. The particular implementation of the job used in this study (WATFIV) is shown in **Fig. 1.**

The synthetic job designed for Batch jobs is somewhat more complex than the one designed for Autobatch jobs. Four parameters: NITER, NOUT, NTAP and NDIS are specified to control the resource usage. NITER controls the number of times the 'compute' loop is executed, NOUT controls how many lines of output are produced, NTAP controls how many records are read from a tape file, and NDIS controls how many records are read from a disc file.

The first task accomplished is to establish the loop control parameter LIMIT = Maximum {NITER, NOUT, NTAP, NDIS}. Within the main loop a pseudorandom number is produced. In addition, the first NOUT times through the loop a line is output; the first NTAP times through the loop a record is read from the tape file; the first NDIS times through the loop a record is read from the disc file; and the first NITER times through the loop a computer routine is invoked. The compute routine involves filling two 5 × 5 matrices with random numbers and then calling a routine to multiply the two matrices to form a third 5 × 5 product matrix. The appropriate settings for the parameters to produce a given demand pattern can be determined from predictor equations established in an earlier section. The particular implementation of the job used in this study (PL/I) is shown in **Fig. 2.**

References

AGRAWALA, A. K., MOHR, J. M. and BRYANT, R. M. (1976). An approach to the Workload Characterization Problem, *Computer*, Vol. 9 No. 6, pp. 18-32.

BUCHHOLZ, W. (1969). A Synthetic Job for Measuring System Performance, *IBM Sys. J.*, Vol. 8 No. 4, pp. 309-318.

COFFMAN, E. G. Jr. and DENNING, P. J. (1973). *Operating Systems Theory*, Englewood Cliffs: Prentice-Hall.

DRAPER, N. R. and SMITH, H. (1966). *Applied Regression Analysis*, New York: John Wiley and Sons.

FERRARI, D. (1978). *Computer System Performance Evaluation*, Englewood Cliffs: Prentice-Hall.

GRENANDER, U. and TSAO, R. T. (1972). Quantitative Methods for Evaluating Computer System Performance: A Review and Proposals, in *Statistical Computer Performance Evaluation* (Freiberger, W., Ed.), New York: Academic Press, pp. 3-25.

HICKS, C. R. (1973). *Fundamental Concepts in the Design of Experiments*, New York: Holt, Rinehart and Winston.

JOHNSON, L. A. (1977). Validation—All Important in Benchmarking, *Proc. CPEUG, National Bureau of Standards Special Publication*, 500-18, pp. 75-83.

LUCAS, H. C. (1971). Performance Evaluation and Monitoring, *ACM Computing Surveys*, Vol. 3 No. 3, pp. 79-91.

SPIRN, J. R. (1977). *Program Behaviour: Models and Measurements*, New York: Elsevier North-Holland.

SCREENIVASAN, K. and KLEINMAN, A. J. (1974). On the Construction of a Representative Synthetic Workload, *CACM* Vol. 17 No. 3, pp. 127-133.

Books reviewed in this issue