Cambridge Scientific Center

36. Y13
July 1966

IBM
Data Processing Division

An Experimental Comparison of Time-Sharing and Batch-Processing

An Experimental Comparison
of
Time-Sharing and Batch-Processing

M. Schatzoff, R. Tsao, R. Wiig

IBM Cambridge Scientific Center Report

International Business Machines Corporation
Cambridge Scientific Center
Cambridge, Massachusetts

July, 1966

36.Y13
July, 1966
Scientific Center Report
Limited Distribution

AN EXPERIMENTAL COMPARISON OF TIME-SHARING AND BATCH-PROCESSING

M. Schatzoff, R. Tsao, R. Wiig

International Business Machines Corporation
Cambridge Scientific Center
Cambridge, Massachusetts

ABSTRACT

The effectiveness for program development of the M.I.T. Compatible Time-Sharing System was compared with that of the IBM IBSYS batch-processing system by means of a statistically designed experiment. An identical set of four programming problems was assigned to each of a group of four programming subjects. Influences external to the systems such as the sequence of problem solution, and programmer and problem characteristics were specified as design factors in the experiment. Data was obtained for six variables (e.g. programmer time, computer time, elapsed time, etc.) which were considered to be definitive of "system effectiveness", and analysis of variance techniques were employed to estimate system differences in these variables after differences due to the design factors had been eliminated. Statistical analysis of the experimental results provided strong evidence of important system differences, as well as a critique of the experimental design itself with implications for further experimentation.

Index Terms for the IBM Subject Index

Computing Evaluation
Time-Sharing---Batch-Processing
07-Computers

LIMITED DISTRIBUTION NOTICE

TABLE OF CONTENTS

I. INTRODUCTION 1

II. EXPERIMENTAL DESIGN 3

III. ANALYSIS AND INTERPRETATION OF RESULTS . . . 7

IV. IMPLICATIONS FOR THE DESIGN OF FUTURE EXPERIMENTS 14

ACKNOWLEDGEMENTS 18

REFERENCES 19

I. INTRODUCTION

Inasmuch as the multiplicity of operational time-shared computing systems has long since dispelled any doubts of their feasibility, time-sharing research is now largely centered upon development of techniques for increasing the utility and effectiveness of such systems. Concomitant with this developmental research effort is an evaluative problem of obtaining measures of the effectiveness of such systems in the problem solving context. The task is hampered not only by the inherent vagueness and ambiguity of the intuitive notion of effectiveness, but by the apparent lack as well of a readily available technique for its measurement. For what is required is not only a measure of the efficiency of the programming system _per se_, but rather a measure, in addition, of effectiveness of the total man-machine interaction. In order to achieve this result, observations and measurements must include the performance and behavior of the individual user and the conditions of his activity.

Moreover, time-sharing lends itself to the facilitation of a variety of computer applications, some of which require so high a degree of interaction with a large scale computer as to render them infeasible except under time-sharing. In order to obtain an effective measure it is therefore necessary to limit the context of inquiry and to focus upon a specifiable application for the evaluation. Clearly the most general application and one that must be accomplished efficiently is that of _program development_, and an initial investigation

is appropriately restricted to this context. But there remains the need also for specification of a standard of comparison, the obvious choice being batch-processing.

The search for a resolution to this problem led to the choice of an operational definition of "effectiveness" in terms of various proposed measurements. Such measurements will obviously be influenced by effects external to the systems such as programmer aptitude, learning and problem characteristics. A statistically designed experiment was therefore constructed in order to isolate these effects and thereby provide meaningful comparisons of the relative effectiveness of a time-sharing system with that of a more customary system of batch-processing. Successful isolation of external effects can reduce statistical variability sufficiently to permit attainment of a given level of precision with a much smaller sample than would have been required otherwise.

It can be useful and informative to carry on this type of investigation even with currently operating specially designed, time-sharing systems that bear a cost penalty for supplementary special equipment needed to adapt, for time-sharing, processors designed for sequential batch-processing. Such evaluations can provide guidance for later application of similar experimental techniques to more advanced systems, which might be expected to exhibit better cost-performance characteristics than the specially adapted processor used in this study. Moreover, the utility of the system in terms of the facilities provided for the programmer, need

in no way be diminished by implementation in current equipment; ... "the essence of a useful time-sharing system lies in the programming, i.e., in the software, and not in the hardware." [1]

A controlled experiment was conducted in the late summer of 1965, using a typical batch-processing scientific computing system (IBM 7094-2 IBSYS) and a flexible time-sharing system providing production applications (the M.I.T. Compatible Time-Sharing System for the IBM 7094-1.). Four programming subjects were selected from technically trained undergraduate students with high programming aptitude. Each individual was assigned an identical set of four problems, two to be coded under time-sharing and two under batch-processing. The four assigned problems were typical of library or system subroutines involving development, implementation, and testing of programs.[1] All subjects had some prior programming experience and received a review of IBM 7094 batch-processing techniques, a brief orientation on usage of the IBM 1050 console with the Compatible Time-Sharing System (CTSS) and a summary of the command language for that system.

II. EXPERIMENTAL DESIGN

Comparisons of system effectiveness for any two computing systems are complicated by numerous factors the effects of which

---

1. Two of the problems were largely numerical, one involving Monte Carlo integration and another, algebraic sorting. The other two problems were essentially of a logical nature, one of them an English to Pig Latin translator and the other a text format conversion.

are difficult to identify and to measure. To start with, the definition of "system effectiveness" is itself open to debate, so that a number of possible measurements relating to this loosely defined concept have been considered, viz:

1. Elapsed time - total working days from start to completion of each problem.
2. Analysis time - total time in minutes spent by each programmer in programming, analysis and debugging of each problem.
3. Programmer's time - total time in minutes spent by each programmer on each problem. This includes analysis time plus such items as keypunching and console time.
4. Computer time - total computer time in minutes for each problem.
5. Number of compilations - number of attempted compilations for each problem solution.
6. Total cost[2] - cost in dollars, for programmer and equipment times, required for each problem solution.

Having settled upon these measures or response variables as useful indicators for comparing the two computational techniques

---

2. Cost estimates in the experiment were based upon somewhat idealized systems which included in both the batch and time-sharing operation only that equipment required to provide the level of service afforded to the programming subjects during the experiment, and omitted any actual equipment that served only a highly specialized or experimental function. Cost data was derived from computer rental and programmer salary estimates; overhead costs were disregarded for both systems.

it is essential to try to eliminate the effects of external factors the influence of which might be so great as to obscure the comparisons of primary interest, namely those pertaining to the various proposed measures of system effectiveness.

In considering the types of measures employed in this study, such as computer and programmer time expenditures, it is immediately apparent that these can be directly influenced by differences in individual programmers and the particular choice of problems. Additionally, since there could be a learning effect from the programming of one problem to the next, the order of problem handling within each system might also be a relevant factor. In order to estimate the system effect differences, independently of differences in the aforementioned factors (i.e., individuals, problems, and order), a modified Graeco-Latin Square design[3] was adopted. The layout of this particular design is shown in Table II-1.

Examination of the design reveals that each programmer coded the same set of four problems, two under time-sharing and two under batch-processing. Furthermore, each problem was coded twice under each system and the sequential order of problem handling by each programmer was different, so that each problem was the first coded by one programmer, the second coded by another, etc. Each problem was completed before the next was begun.

---

3. See [2] for discussion of Graeco-Latin Squares.

TABLE II-1

Experimental Design

		Problems 1	2	3	4
Programmers	1	T_2	B_1	T_3	B_4
	2	B_1	T_2	B_4	T_3
	3	B_4	T_3	B_1	T_2
	4	T_3	B_4	T_2	B_1

NOTATION: B denotes batch-processing

T denotes time-sharing

The subscripts denote the sequence of problem handling for each programmer.

It should be noted that the design is orthogonal with respect to the main effects, which are assumed to be additive. Thus, the design permits independent estimation of all the main effects (i.e., effects due to differences in systems, programmers, problems and order of problem solution within system).

III. ANALYSIS AND INTERPRETATION OF RESULTS

The observations obtained from the experiment are shown in Table III-1. Summaries of these results for each of the design factors are given in Table III-2, together with the observed significance levels as calculated in the analysis of variance[4]. An observed significance level is the probability of observing an F value as large as or larger than the one computed if there is no difference in the response variable with regard to the design factor. Thus, a very small observed significance level would cast doubt upon the hypothesis of no difference in the response variable due to the particular design factor; for example, referring to Table III-2, the observed difference in programmer's time for the two system (i.e., 5672 minutes for time-sharings vs. 2737 minutes for batch) may be considered indicative of a basic difference in the systems, since the observed significance level is only .019.

Similarly, the number of attempted compilations (118 for time-sharing vs. 49 for batch) appears to be significantly different

---

4. See [2] for discussion of the analysis of variance.

TABLE III-1

Experimental Results

Programmers	Problems:	1	2	3	4
		T_2	B_1	T_3	B_4
1	Elapsed Time(Days)	6.5	4.0	10.0	9.0
	Analysis Time (Min.)	450	295	915	420
	Programmer Time (Min.)	810	355	1250	533
	Computer Time (Min.)	23.3	9.6	21.9	13.5
	No. of Compilations	38	4	26	8
	Total Cost(Dollars)	368.52	107.35	370.81	152.65
		B_1	T_2	B_4	T_3
2	Elapsed Time(Days)	4.0	0.5	8.0	3.0
	Analysis Time (Min.)	152	60	95	300
	Programmer Time(Min.)	195	75	115	355
	Computer Time(Min.)	16.2	0.7	10.8	3.6
	No. of Compilations	6	1	6	4
	Total Cost(Dollars)	160.94	13.60	106.55	68.43
		B_4	T_3	B_1	T_2
3	Elapsed Time(Days)	3.0	0.5	5.0	3.0
	Analysis Time (Min.)	310	60	486	550
	Programmer Time(Min.)	340	145	537	890
	Computer Time(Min.)	6.0	3.0	25.2	12.1
	No. of Compilations	3	2	7	20
	Total Cost(Dollars)	73.00	49.48	262.04	214.84
		T_3	B_4	T_2	B_1
4	Elapsed Time(Days)	4.0	5.0	2.0	8.0
	Analysis Time(Min.)	563	95	161	442
	Programmer Time(Min.)	1369	110	778	552
	Computer Time(Min.)	13.7	8.4	13.7	10.8
	No. of Compilations	13	4	14	11
	Total Cost(Dollars)	261.32	83.90	231.77	128.40

T_i denotes that the programmer's i^{th} problem was handled under time-sharing.

B_i denotes that the programmer's i^{th} problem was handled under batch-processing.

TABLE III-2

Summary of Experimental Results by Design Factor

α = observed signficance level

1. Systems

	Time-sharing	Batch	F	α
Elapsed Time	29.5	46	4.40	.081
Analysis Time	3059	2295	1.08	>.200
Programmer Time	5672	2737	10.04	.019
Computer Time	92	100.5	<1	>.200
No. of Compilations	118	49	6.23	.047
Total Cost	1578.77	1074.83	4.34	.082

2. Order Within Systems

	Time-sharing				Batch			
	1st	2nd	F	α	1st	2nd	F	α
Elapsed Time	12.0	17.5	< 1	>.200	21.0	25.0	< 1	>.200
Analysis Time	1221	1838	1.41	>.200	1375	920	< 1	>.200
Programmer Time	2553	3119	< 1	>.200	1639	1098	< 1	>.200
Computer Time	49.8	42.2	< 1	>.200	61.8	38.7	2.66	.154
No. of Compilations	73	45	2.05	>.200	28	21	< 1	>.200
Total Cost	828.73	750.04	< 1	>.200	658.73	416.10	2.01	>.200

3. Programmers

	1	2	3	4	F	α
Elapsed Time	29.5	15.5	11.5	19.0	3.85	.075
Analysis Time	2080	607	1406	1261	2.70	.139
Programmer Time	2948	740	1912	2809	4.89	.047
Computer Time	68.3	31.3	46.3	46.6	2.31	.176
No. of Compilations	76	17	32	42	3.28	.101
Total Cost	999.33	349.52	599.36	705.39	4.95	.046

4. Problems

	1	2	3	4	F	α
Elapsed Time	17.5	10.0	25.0	23.0	2.91	.123
Analysis Time	1475	510	1657	1712	2.33	.174
Programmer Time	2714	685	2680	2330	4.30	.061
Computer Time	59.2	21.7	71.6	40.0	4.78	.050
No. of Compilations	60	11	53	43	2.46	.160
Total Cost	863.78	254.33	971.17	564.32	7.10	.021

for the two systems at the .047 observed significance level. Additionally, somewhat higher significance levels of .08, corresponding to system differences in elapsed time (50% higher for batch-processing) and total cost (50% higher for time-sharing), were observed. It did not appear that there were any significant system differences with respect to computer time or analysis time. It should be noted that the experiment was designed in such a way that comparisons of these two systems are independent of any effects which might be attributable to the other design factors, namely programmers, problems and order. As we shall see, some of these effects were so large that the system differences might have been disguised had the experimental design not allowed for their isolation.

Further examination of Table III-2 facilitates identification of the other design factors which appear to effect significant differences upon one or more of the response variables, as judged by their accompanying observed signficance levels. For example, differences in total cost (as great as 3 to 1) and programmer time (as great as 4 to 1) among the different programmers appear to be significant, despite the fact that all of the programmers had similar formal technical undergraduate backgrounds, and each received an A grade on the IBM Data Processing Aptitude Test. Table III-2 also reveals large and apparently significant differences in programmer's time, computer time and total cost due to the effect of the different problems. The order of processing problems on each system had no apparent effect upon any of the response variables.

As might be expected, the six reponse variables chosen for this experiment are not independent, and hence the observed significance levels for the six variables are not independent. The interdependencies of the response variables are summarized in Table III-3, which is the matrix of partial correlation coefficients for the six variables after eliminating the effects due to the design factors. For example, it is interesting to note that the correlation coefficient between programmer time and computer time, after eliminating the effects of differences in programmers, problems, systems and order from both, is only .18.

Among the virtues of a time-sharing system is the availability of selective console debugging techniques, and for this an elaborate battery of diagnostic tools have been developed. But these techniques are truly available only to one already trained in their use. Our subjects, lacking such facilities, were constrained to employ under time-sharing the same habits that had been evolved effectively to cope with batch-processing operating conditions, i.e., desk debugging, recompilation, and repeated execution. Moreover, the very availability of the time-sharing console makes for the likelihood of abuse in this mode of operation for there is far less constraint to correct at any time a maximum of programming blunders when the opportunity for immediate compilation and test is always present.

Indeed, as shown in Table III-4, the number of compilations under time-sharing was more than double that experienced

TABLE III-3

Partial Correlations Among Response Variables, After Eliminating the Effect of the Design Factors

	Elapsed Time	Analysis Time	Programmer Time	Computer Time	No. of Compilations	Total Cost
Elapsed Time	1.00	.54	.23	.53	.83	.64
Analysis Time	.54	1.00	.80	.28	.30	.49
Programmer Time	.23	.80	1.00	.18	-.01	.38
Computer Time	.53	.28	.18	1.00	.60	.95
No. of Compilations	.83	.30	-.01	.60	1.00	.72
Total Cost	.64	.49	.38	.95	.72	1.00

TABLE III-4

Comparison of Two Systems

	Time-Sharing	Batch-Processing	T/B
Computer Time (minutes)	92	100.5	.92
Number of Compilations	118	49	2.41
Computer Time/ Compilations (minutes)	.78	2.05	.38
Cost/Compilation (Dollar)	13	22	.59
Programmer's Time/Compilation (minutes)	48	56	.86

under batch-processing. Thus, normal program debugging techniques seem to be wasteful under time-sharing for they apparently result in excessive compilations. However, the system efficiency of CTSS seemed sufficient to compensate for this increase, since the computer time per compilation under CTSS was only 38% as great as that experienced under the batch system.

In comparing computer time for the two systems, it should be noted that the time-sharing system is implemented on a 2 microsecond cycle 7094-1, while the batch-processing system is implemented on a 1.4 microsecond 7094-2. Furthermore, the time-sharing system does not utilize dynamic relocation techniques, so that memory must be continually reconstituted for each user processing cycle.

IV. IMPLICATIONS FOR THE DESIGN OF FUTURE EXPERIMENTS

Scientific endeavor is essentially an iterative process involving experimentation, observation and continual re-evaluation of hypotheses based on accumulated experience. Information acquired at

particular phases of this process provides bases for directing the course of subsequent phases. Thus, results from this initial small-scale experiment, limited in scope to the comparative assessment of two particular systems, bear not only upon the measures themselves of system effectiveness, but apply with equal validity to a critique of the general assumptions upon which the experiment was based. A number of useful observations can thus be made concerning the design of future experiments of this general nature.

1. The variation attributable to problem and programmer differences (cf. Table III-2) is of sufficient magnitude to suggest inclusion of these factors in the design of future experiments in order to separate such effects from the system characteristics of interest.

2. The learning effect, as measured by the variation due to order of processing the different problems, appeared to be negligible in the experiment relative to the other factors being measured. One might however anticipate that under altered circumstances (e. g. with an enlarged sample size) the learning effect might indeed become relevant. An alternative is then to randomize the order of problem solution rather than to consider order as a separate factor in future experiments. Advantages of randomization over inclusion as a design factor are a greater flexibility in design and a larger degree of freedom for estimating the error variance.

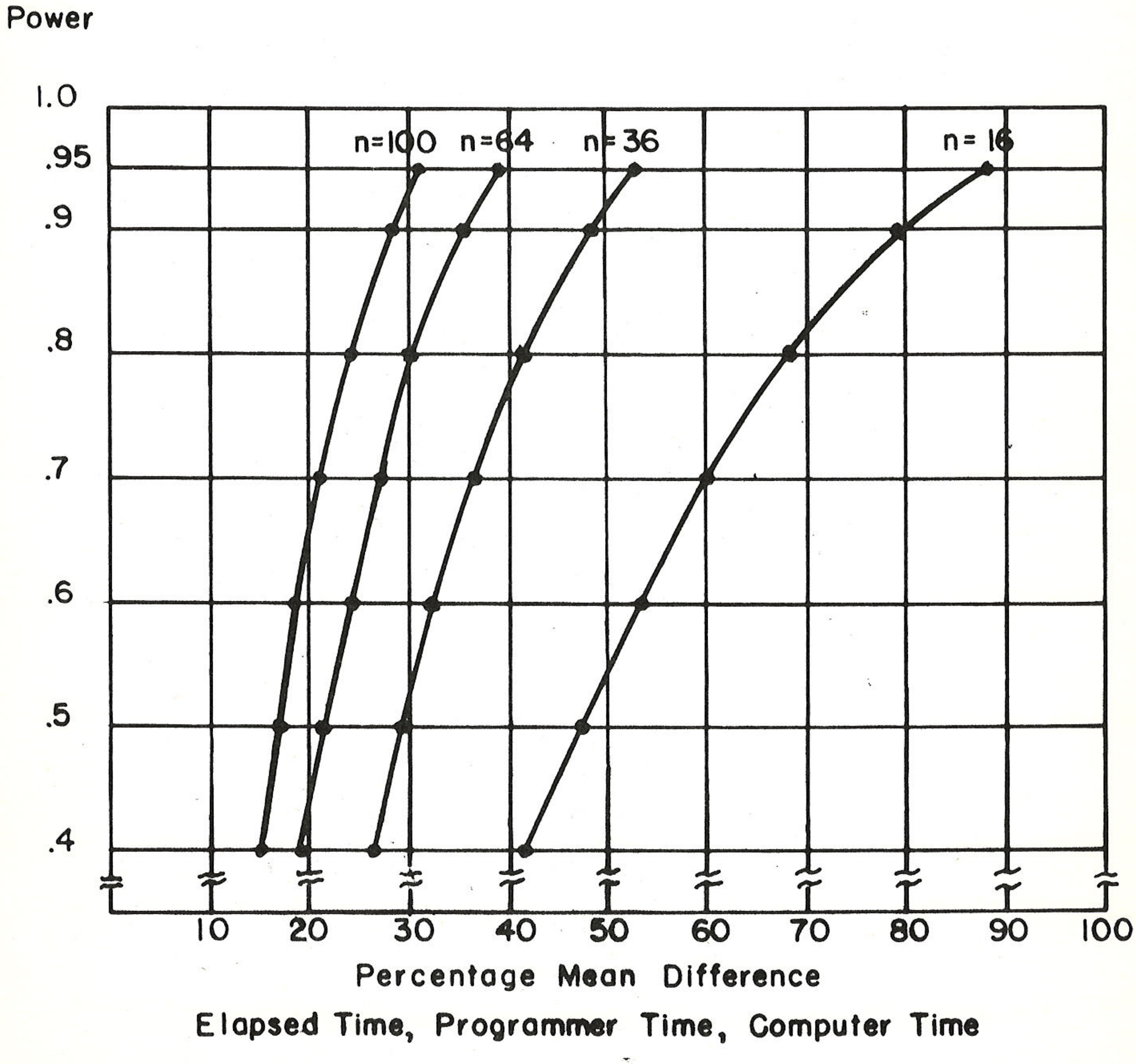
3. A critical question in the planning of an experiment is determination of sample size. Our experiment may appear to be of

small scale; however, under the hypothesis that there are no actual system differences, the observed significance level reflects the actual sample size used for the particular experimental design. If there are indeed differences between the systems, a question arises as to the sensitivity of the experiment for detecting such differences. In this context questions of sample size become relevant. One possible index of sensitivity can be obtained from power curves, which give the probabilities of observing various significance levels as functions of variance, true difference in mean responses and sample size.

Our initial experiment provides us with an estimate of the variance for each response variable and enables the derivation of power curves for them. In our experiment the response variables elapsed time, programmer time, and computer time, exhibit almost identical sample coefficients of variation and therefore the same set of power curves are applicable to all three. For example, Figure IV-1 shows a set of power curves based on significance levels of .05 for these responses. The abscissa shows the mean difference in the response between systems, expressed as a percentage of the mean response for both systems. The ordinate indicates the power or probability of observing significance levels as small as .05. Thus, we see that in our experiment the probability of detecting differences of 40% at the .05 significance level, was only about .4. Increasing the experiment to 6 x 6 (i.e., 6 programmers each solving 6 problems)

Figure IV - 1

Power Curves at .05 Level
For Various Sample Sizes (n)



would double the probability of detecting differences of 40% and would enable us to detect differences as small as 30% with probability one-half. Although enlargement of sample size increases sensitivity, the limiting factor is usually economic in nature, so what is sought is a trade-off involving allocation of resources and the attainment of prescribed levels of sensitivity.

The power curves depicted in Figure IV-1 thus provide a basis for deciding how to allocate resources to further experiments of this nature; they provide also a measure of how well we have done statistically in the initial experiment.

ACKNOWLEDGEMENT

The authors wish to acknowledge the generous assistance of the Massachusetts Institute of Technology in providing access to computing facilities at the M. I. T. Lincoln Laboratory and at the M. I. T. Computation Center during the course of the experiment.

REFERENCES

1. Crisman, P. A., (Ed.). The Compatible Time-Sharing System, A Programmer's Guide. Preface to the Second Edition by F. J. Corbato, p l. The M.I.T. Press, Cambridge, Massachusetts, 1965.

2. Kempthorne, O. The Design and Analysis of Experiments. John Wiley & Sons, New York, 1952.

IBM®