1968

Numerical Analysis for the Consumer

D. Dodson

J. Ewing

P. Miller

W. Nylin

E. Petarik

See next page for additional authors

Report Number:

68-026

Authors
D. Dodson, J. Ewing, P. Miller, W. Nylin, E. Petarik, and S. Prues

NUMERICAL ANALYSIS FOR THE CONSUMER

by

D.Dodson, J.Ewing, P.Miller
W.Nylin, E.Pek[illegible]rek, W.Porter, S.Pruess

with

FOREWARD by J.R.Rice

CSD TR 26
September 1, 1968

TABLE OF CONTENTS

FOREWARD

The objective of this report is to present a broad discussion of the actual application of numerical analysis. Attention is focused on the "typical" user - the person with mathematical problems who needs answers. The most difficult problems are:
i) to identify this "typical" consumer of numerical analysis
ii) to identify his sources of numerical analysis information
iii) to identify his desires and his needs for numerical analysis.
These problems do not yield to mathematical analysis and are rarely considered by numerical analysts. Yet if one believes that numerical analysis is to be used rather than contemplated, one cannot ignore these questions. It is certainly clear that there is a large communication gap between the professional numerical analyst and the "typical" consumer - whoever he is. It is not clear that this gap is currently narrowing. The emphasis in this report is on approaches to narrow this gap which exploit computers and computing systems.

This material was originally gathered in the spring of 1968 by the members of the "Graduate Seminar". Since the topic was covered in about six weeks, there is a lack of depth at many points of the analysis. Nevertheless, I feel that a reasonably accurate (if not too well focused) picture of the situation is presented here.

This seminar was under my direction and hence the material inevitably reflects my prejudices. Furthermore, I have edited (and shortened somewhat) the original material. I have tried to preserve the authors original views during this process-- even at those points where I consider the views to be incorrect.

John R. Rice

NUMERICAL ANALYSIS FOR THE CONSUMER

by

D. Dodson, J. Ewing, P. Miller, W. Nylin, E. Pekarik,

W. Porter, S. Pruess

1. WHO ARE THE USERS OF NUMERICAL ANALYSIS?

To answer the question as to who are the consumers of numerical analysis one must first decide what constitutes a consumer of numerical analysis. Henceforth, a consumer of numerical analysis is referred to only as a consumer. A consumer is any person or group of people who directly uses numerical analysis or methods of numerical analysis from a theoretical or computational standpoint. Also, a potential consumer is any person or group of people who have a use for numerical analysis in their work (whether they realize it or not).

From the above definitions, a consumer or potential consumer could be a company, a particular occupation, a group of people or an individual person. It therefore seems reasonable to search for consumers in each of the above categories.

First consider which individuals are consumers. There are so many potential consumers who are not yet consumers, some care must be taken not to include them. Individuals who contribute to the development of numerical analysis by publishing can be found by searching journals for such contributions. People who use numerical analysis as a tool for solving other problems and publish their results can also be found by searching literature. By sampling the literature in the library, it would be difficult to obtain a large number of people in this manner, although there seems to be literature in almost any field exhibiting a use for numerical analysis. Other people who are consumers of numerical analysis are the subscribers to publications in the field. The subscription list to these publications would yield a list of potential consumers, all of whom might not be actual consumers. Subscription lists to some of these publications are available. Individuals who attend conferences on numerical analysis or use the numerical routines from organizations like Share are consumers. Still another way to find consumers is to determine users of numerical analysis routines at computing centers, etc.

There is still a large group of consumers who might not be found by any of these methods, since they write their own numerical routines. To find these people it is necessary to determine how they received their training in numerical analysis.

From this one is led to examine the group of people who have in some sense received formal instruction in numerical analysis. Thus the people who have attended courses in numerical analysis are at least potential consumers, if not actual consumers. The only other way to determine if an individual is a consumer of numerical analysis is to ask him.

Once it is known which people are consumers, one can determine which occupations are consumers. To determine what occupations are consumers or potential consumers of numerical analysis, a questionnaire was distributed to people in various fields at Purdue to determine the extent of useage of numerical analysis methods in their field and thus try to determine if that field is a consumer. This is a relative measure since almost every occupation contains some consumers or potential consumers.

To determine which companies are consumers, one can examine its employees to determine the number of individual consumers or the extent to which they employ people of an occupation that is a consumer. The extent to which a company uses a computer, participates in organizations like Share, has projects using numerical analysis, and has periodicals and books on numerical analysis in the library all help to determine if a company is a consumer.

Another important question is what do the consumers want in the way of numerical analysis. The questionnaire mentioned earlier is also designed to help answer this question.

Approximately 2500 questionnaires were passed out to graduate students and faculty members in 24 departments outside the Division of Mathematical Sciences. The survey questionnaire is as follows:

COMPUTER SCIENCE QUESTIONNAIRE

The purpose of this questionnaire is to determine to what extent different departments use numerical analysis or numerical methods on a computer and to suggest how the Computer Science Center can better serve the needs of each department in this area.

Your co-operation in compiling this survey will be appreciated.

Department ______________________ Position ______________

Area of Specialization ______________________

1. Have you used a computer or computer results generated for you in your work? Yes ____ No ____

2. Do you have an intuitive idea of what using numerical methods or methods of numerical analysis means? Yes ____ No ____

3. Do you have or do you know of problems in your field which are being solved or could be solved using numerical methods on a computer? Yes ____ No ____

4. Have you used such methods to help solve a problem in your field? Yes ____ No ____

5. Would you have a use for these methods if somehow they were more readily available or easier to use? Yes ____ No ____

6. How would you rate your knowledge as to how to use a computer and as to what is available for usage on a computer? Good ___ Average ___ Poor ___

7. Have you used any of the routines made available by the Computer Center? Yes ____ No ____

8. With regard to these routines do you want
   A. a detailed description of the algorithms used and how they work? Yes ____ No ____
   B. only an easy way to use them? Yes ____ No ____

9. How do you rate the desirability of the following features in the implementation of numerical methods on a computer?

	High	Medium	Low
A. Economical execution	____	____	____
B. Convenient to use	____	____	____
C. Very high reliability	____	____	____
D. Extensive diagnostics in case of trouble	____	____	____
E. Error bounds on computation	____	____	____
F. Conversational use so users can make execution time decisions	____	____	____

10. Are there any numerical methods or routines presently not available which you would like to see the Computer Science Center make available?

____

____

11. Are there at present any routines with which you are dissatisfied and why?

____

____

12. Are there any routines available on the IBM 7094 which you would like to see made available on the CDC 6500? ______________________

13. Comments ______________________

The tabulation of the results of the questionnaire is based on 395 respondants of whom 49% are faculty members, 47% are graduate students, and 4% are bashful.

On the first question, 88% answered yes and 12% answered no. It was expected that responses would be heavily weighted toward computer users, but 12% of the total, or 47 non-users, is a significant group.

On question 2; 92% answered yes, 6% answered no, and 2% did not answer. This indicates that a high percentage of people have some knowledge of numerical analysis and can reliably answer questions pertaining to their usage of it.

70% of the responses were yes to question 3; 3% were no, and 27% declined to answer. The fact that only 3% of the answers were definitely no, combined with the fact that there was at least one affirmative answer in each field, suggests that every field interviewed is to some degree a potential consumer.

On the fourth question 49% of the responses were yes, 25% were no, and 26% declined to answer. This question shows to what degree different fields are consumers relative to each other.

With regard to the fifth question 79% of the responses were yes, 7% were no, and 14% did not answer. Since 70% of the respondants said they would have a use for some numerical methods, if they were more readily available or easier to use, seems to point out a tremendously large vacuum of routines using numerical methods already written for application to specific problems in each field. These routines may have been written somewhere, but not widely publicized or compatible between computer installations. It should be the responsibility of the universities and the government to promote the standardization of computer languages and the publication of available routines. 6% of the respondants said they have never used a computer, answered yes to question 5 and 12% answered yes on question 3, no on question 4 and yes on question 5. These statistics help to substantiate the need for the compatibility of languages and distribution of specific routines.

The results on question 6 were scaled between 0 and 100, with 0 and 100 representing poor and good respectively. The average index of the responses is 53. Thus the respondants considered themselves to be an average group of computer users.

63% of the respondants have used routines made available by the computing center, 36% have not and 1% evidently were not sure. Since only 12% have not used a computer, that leaves 24% of the total who have not used the computing center routines. Their reasons were mainly preferring to write their own and not knowing what is available from the center.

It is surprising to find on question 8 that 58% wanted a detailed description of the algorithms used with routine explanations and 34% wanted only an easy way to use them. A large number of the latter only wanted a reference to the methods used for possible investigation. Consumers appear to want more than canned routines. They are also interested in the reasoning behind the various methods. Question 9 attempts to find out what consumers really want from computer implemented numerical methods. Again the results are scaled from 0 to 100, with 0 and 100 representing low and high desirability, respectively. The average index is listed below for the total responses.

1.	Convenient to use	88
2.	Very high reliability	85
3.	Economical execution	67
4.	Extensive diagnostics	67
5.	Error bounds on computation	61
6.	Conversational use	43

It is interesting to note how little importance is placed on conversational use. The results on this question are very consistent between the different departments. A more complete tabulation of the results follows.

TABLE 1: Breakdown by Department of Responses to Question 9.

	Department	Responses	Index (from 0 to 100) 9-A	9-B	9-C	9-D	9-E	9-F
A.	A.A. & E.S.	19	67	87	97	84	64	39
B.	Ag. Econ. & Stat.	31	93	86	80	70	59	44
C.	Ag. Engineering	15	68	94	73	61	67	57
D.	Bio-Chemistry	11	56	88	81	67	80	50
E.	Bio-Nucleonics	8	58	83	100	83	59	50
F.	Botony & Plant Path.	5	63	90	70	90	50	38
G.	Chemical Eng.	17	53	88	89	74	54	30
H.	Chemistry	6	50	100	100	100	67	50
I.	Civil Eng.	37	53	88	81	59	55	39
J.	Economics	2	50	100	75	100	75	50
K.	Electrical Eng.	53	59	90	77	64	57	42
L.	Forrestry & Cons.	14	67	100	88	75	55	36
M.	Geo-Sciences	1	0	100	100	100	50	0
N.	Horticulture	5	100	100	88	67	75	33
O.	Industrial Adm.	2	50	100	100	50	0	0
P.	Industrial Eng.	16	52	84	75	81	48	44
Q.	Industrial Science	6	40	90	75	84	50	50
R.	Material Science	5	63	50	63	50	75	33
S.	Mechanical Eng.	43	78	90	89	68	72	48
T.	Nuclear Eng.	24	71	84	90	71	54	52
U.	Pharmacology	12	65	93	86	50	54	66
U.	Pharmacology	12	65	93	86	50	54	66
V.	Physics	34	70	91	95	69	60	40
W.	Psychology	12	58	80	92	64	64	36
X.	Sociology	15	75	79	86	58	68	38

TABLE 2. A Breakdown of Questions 1 through 7 by Department. The Numbers are all in Percent of the Total Responses from a Department. The Letters are Associated with the Departments in Table 1.

	1.		2.		3.		4.		5.		6.			7.	
Dept.	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Hi.	Med.	Low	Yes	No
A.	97	0	100	0	100	0	84	11	74	11	58	32	11	58	37
B.	97	0	97	0	97	0	97	0	71	3	26	52	16	65	32
C.	100	0	93	7	93	7	80	20	87	0	27	6	7	80	20
D.	18	82	82	9	91	9	18	82	72	9	0	27	73	0	91
E.	62	38	75	25	75	25	67	38	88	0	0	38	50	50	50
F.	40	60	60	40	100	0	40	60	40	60	0	40	60	20	80
G.	94	6	100	0	100	0	88	12	94	0	29	53	18	88	6
H.	67	33	83	17	83	0	50	50	50	33	17	33	50	67	33
I.	100	0	100	0	100	0	100	0	100	0	100	0	0	50	50
J.	100	0	100	0	100	0	0	100	0	100	0	100	0	100	0
K.	80	20	60	0	80	0	40	40	40	20	0	20	80	60	40
L.	100	0	100	0	100	0	100	0	100	0	0	50	50	100	0
M.	40	60	100	0	100	0	0	100	60	40	0	40	60	20	60
N.	97	3	95	5	97	3	81	19	89	5	27	54	19	73	27
O.	95	5	98	2	98	2	83	17	79	4	32	47	21	72	26
P.	86	7	86	7	93	0	79	14	72	14	36	50	14	93	7
Q.	43	57	72	28	86	7	36	64	86	7	14	50	36	21	72
R.	95	5	98	2	100	0	91	9	77	2	37	49	14	68	30
S.	100	0	83	17	83	17	67	33	100	0	17	83	0	67	33
T.	96	0	96	0	96	0	84	12	88	0	50	34	12	75	21
U.	85	15	97	3	97	3	65	35	76	12	41	41	15	41	59
V.	92	8	100	0	92	8	67	33	92	8	42	25	33	67	33
W.	87	13	87	13	100	0	80	20	93	7	7	40	53	60	40
X.	88	12	82	12	88	12	56	44	56	19	38	44	12	69	19

2. HOW PEOPLE OBTAIN NUMERICAL ANALYSIS INFORMATION

Since very little, if any, information has been published on how people actually obtain numerical analysis information, it is only possible to surmise how this is or might be done based on personal experience and indirect evidence.

2.1 Formal Education. Initial introduction to numerical analysis information may be through formal education. Almost all universities offer some courses in numerical analysis, either in the department of mathematics or computer science.

For those persons who are not students, there are a number of sources of formal and semiformal instruction available. Several universities offer a number of short courses which cover a variety of topics in computer science and related areas. The Association for Computing Machinery has also recently begun to sponsor a number of several-day seminars and tutorials on topics of current interest in computer science. A significant source of information within a company are short courses conducted to keep personnel acquainted with current developments.

2.2 Text and Reference Books. For information beyond that available in coursework, a number of textbooks and other references on numerical analysis are available. Several useful collections have appeared--for example, Ralston and Wilf, Methods for Digital Computers. The Ralston and Wilf book summarizes a number of methods and describes algorithms for the solution of a wide variety of problems and provides references to sources where a more detailed development of the various methods can be found.

2.3 Technical Journals. For information on developments more recent than found in books, one can go to the various professional journals. Probably the publications most widely known by persons in computer science in the United States are the publications of the Association for Computing Machinery, although there are a number of other significant publications, e.g., SIAM Journal of Numerical Analysis, Numerische Mathematik, Mathematics of Computation, SIAM Journal of Applied Mathematics and the Computer Journal.

2.4 Communication Among Professionals. For those persons most closely associated with the field of numerical analysis there are several important sources of information which are not usually available to the average user of numerical analysis information. One source is through contacts with other persons working in the field, both in the form of personal communications and in the form of pre-publication copies of manuscripts. Another very important source of information is their own research.

2.5 Consultants. Persons or groups of persons which lack the training or the time to obtain numerical analysis information themselves can hire consultants to help them. Because consultants are also used in cases which only involve "data processing" problems, just how much consulting is done on problems directly related to numerical analysis would be difficult to tell without a detailed investigation into the activities of consultants. That consulting is a source of information that is being utilized is evidenced by the increase in the number of firms specializing in consulting.

2.6 Subroutine Libraries and Interactive Systems. Most computing centers have a number of subroutines available for the solution of certain kinds of numerical problems. These presumably have been used successfully by others and have the advantage of being in machine-useable form. Interactive systems are valuable for some types of numerical problems in that the judgment of the user can be used to advantage rather than requiring the user to specify in advance what action should be taken for each possible situation. The task of the user can be further simplified by languages which minimize the programming effort necessary for solving the most common numerical problems. One approach to this problem is to provide built-in procedures for such problems. The ideal system from the user's viewpoint would be an automatic numerical analysis system which when given a problem would choose an appropriate method and then supply an algorithm for that method.

2.7 The Strengths and Weaknesses of Sources of Information. Consider first formal education and course-work as a source of information. A good teacher who is familiar with the topics of the course can help make up for the deficiencies which seem to be inherent in any published text (unless of course the instructor and text are both deficient in the same areas). A teacher may also be able to provide advice on practicalities of various methods on present computers and call attention to new developments in the field. The basic problem, of course, is that good teachers are not always easy to find. Also a person who spends the major part of his time teaching might not have the time to keep up with new developments in his field, much less be actively engaged in research necessary to put him near the top in his field. New developments make a process of continuous education necessary.

Reference books and professional journals contain much useful information, but there are two major deficiencies: time lag between research and publication, and information retrieval. Those who are most actively engaged in research do not have the time to prepare a book for publication until later. For this reason a book which is written by only one person might not contain the results of any research done during the last ten years. Books which consist of a collection of shorter articles by a number of authors may be more current, but a time lag of five years between research and publication is probably about minimal. The problem of information retrieval is only partly solved by such aids as Computing Reviews, Math Reviews and Computer Abstracts and the various indices that are available for reviews. Books which cover a broad range of topics are not usually indexed under the individual topics. An even more serious problem might be the lack of complete coverage by these reviews. Computer science and numerical analysis are still new enough that the area has not yet been completely defined, making complete coverage by reviewers an impossibility.

Subroutine and program libraries are widely available but not so frequently used. Ideally a subroutine should be representative of the best way known (best for the machine used) for solving some particular problem. A subroutine should be accompanied by a writeup which explains the method used, its justification, any limitations on accuracy, reliability, etc. of the program, and clear instructions for proper use of the subroutine. If this were the case, then one could use such subroutines with confidence. In practice subroutine libraries frequently contain subroutines which are not optimum. Frequently writeups available for a subroutine tend to degenerate into a "cook book" for usage and to avoid explanation or justification of the method used. This leads to the use of the subroutine on problems for which the method may be ill suited. For these reasons a person capable of writing his own subroutines frequently avoids using subroutines written by someone else, or else he uses these routines with reluctance and with the conviction that given the time he could do better.

3. SUBROUTINE LIBRARIES

Subroutine libraries exist for the obvious purpose of saving users the effort of writing their own programs. Though it is easy to list the objections to using library subroutines, the case for such a facility is clearly justified on the basis of the time saved.

The most obvious weakness of the subroutine library is that users must accept the validity of a program on good faith. Certainly, no library is going to claim that all of its routines work exactly as specified. While it is comforting that an author is not going to submit his program to a library until he is confident it works properly, it is naive to expect any routine to work for every case. In addition to this weakness, library routines customarily offer a number of options, which a user might not need for a particular application. Problems in understanding documentation and data format compatibility are objections worthy of mention.

The Purdue Computer Center offers a number of libraries to its users:

(1) The Computer Science Center Subroutine Library includes about 70 routines written locally (in MAP, FAP, and FORTRAN) and/or adapted from SHARE programs. Experience with five of the routines indicates that the performance is as specified though the write-ups are difficult to understand. Experience with the Gauss-Laguerre quadrature routine was disastrous - the weights for $N = 14$ did not add to 1.

(2) PUFFT Subroutine Library is a disc resident FORTRAN library of about 50 routines. That object decks are not available is not a restriction as PUFFT can compile and load a source deck faster than the IBJOB loader can load an object deck. The big restriction is that the routines can be compiled only into PUFFT programs.

Many of the routines are adaptations of (1) though some were written especially for PUFFT by Purdue staff members; the documentation is, in general, better than (1).

One person's experience with this package has been quite satisfactory - the big advantage is that the inconvenience and the errors associated with handling decks are avoided. For example, the Gauss-Laguerre routine (at one time a copy of the corresponding routine in (1)) is correct under PUFFT.

(3) The IBM 360 Scientific Subroutine Package contains 200 FORTRAN routines compatible with the IBM 7094 and CDC 6500. Documentation is contained in comment cards and is quite understandable and uniform.

These subroutines treat matrices in a unique manner - all two dimensional actual arguments are treated as one dimensional variables in the subroutines. This has the disadvantage that the user must either dimension his arrays to fit particular cases or use a special subroutine (in the package) to convert to the correct storage layout. The advantage is that under no circumstances do subroutines need be changed for dimension compatibility.

Casual experience has suggested that this is a reliable set of subroutines.

(4) The 7094 disc library contains about 30 routines accessible through the IBDSK system. Again, there is quite an advantage here in not handling decks. Purdue's CSC Technical Newsletter of October, 1967.

(5) The Statistical Library consists mainly of 1964 BMD series from UCLA. These are complete programs - in fact, the manual claims that a knowledge of programming is not required. The data deck set-ups are clearly explained and illustrated. The department of statistics maintains consultants familiar with these routines.

(6) The CDC 6500 disc library is still in the developmental stage.

The main inter-facility libraries to which Purdue CSC users have access are the SHARE (an organization of IBM users) library and the VIM (an organization of CDC 6000 series users) library. Naturally, the readability of the inter-facility libraries varies according to the author.

The SHARE routines (approximately 2000 in the SHARE 3000 series) are immediately available from tape dumps and master decks. When additions to the library are made, announcements are printed in the Technical Newsletters of the CSC. The programs available locally are coded in MAP, FAP, FORTRAN II, and FORTRAN IV and are for the IBM 709/7090/7094 series.

The VIM routines are not available locally - users must contact the central distributor.

4. INTERACTIVE SYSTEMS

An interactive system is a combination of computer hardware and software which converses with a user at a remote terminal. The system might serve only one user, (a dedicated system) or it may serve many at once, maintaining for each user some impression of being the only user. Interactive systems are basically user centered. Varying levels of communication and degrees of interaction are maintained by different systems, ranging from computer assisted instruction (teaching by the Socratic method) to something like Quicktran, or PUFFT Time Sharing System, toward the other extreme. In any case an interactive system provides a shorter turn-around time, than an ordinary batch processing system does.

Our specific interest is with interactive systems with a mathematical, or numerical analysis, orientation. At this point we are more concerned with what the system should be like, what it should do, rather than with details of implementation.

The most obvious, and very likely the most important factor in the design of an interactive system is the language for communication between man and machine. The primary goal in formulating this language is to minimize the demands on the user. Ideally, the computer would be able to understand the user writing in his own language. This is extremely ambitious; clearly a practical system must impose further restrictions. Since the user must type instructions,simplicity and brevity are important. An excessively verbose language makes typing instructions and waiting for the system to type messages burdensome to the initiated user.

The language should be chosen so that a novice can use the system without much difficulty. This requires a high degree of readability in both the computer's messages to the user and the user's statement of instructions. Ordinarily this means longer messages and more typing. It probably also requires a specially tailored character set if special terms or symbols are to appear frequently. Not all users are novices; after a person has had some experience with the system, shorter comments suffice. An inexperienced user may be satisfied with, and indeed may desire, a communication that runs something like: (computer's part underlined):

WHAT NEXT? PLOT
TYPE INDEPENDENT VARIABLE X
TYPE FUNCTION OF X TO BE PLOTTED. Y
PLEASE DEFINE Y. Y = 2 X + 1
XMIN = 3
XMAX = 10

A more experienced user would probably prefer an exchange in the spirit of the following:

WHAT NEXT? PLOT Y = 2 X + 1, FOR X = 3, 10

The same spirit of varying wordiness carries over to things like error messages, etc.

Three desirable features aimed at this need are:

1. Having several alternate forms for various instructions.
2. Permitting the user to select a message level to provide messages varying from terse to quite detailed and verbose.
3. Implementing an instruction "EXPLAIN" which causes the system to provide further information.

The level of communication desired requires special, more powerful operators and a macro-instruction facility. For a novice in numerical analysis to benefit from the system, he should be able to write, for example, $\int_a^b f(x)dx$ (where f is already defined), or some reasonable equivalent of it, and have the system do the work for him. In all such operations, a reasonable, perhaps user specified, level of accuracy should be maintained. If that accuracy is beyond the system, it should provide an answer together with an estimate of its accuracy.

It is clearly desirable to have two modes of giving instructions to the system: a do-this-now mode, and a procedure definition mode. We note that it is possible to approximate the former with the latter if care is taken in the design. The necessity of the latter for a truly useful language is obvious.

Some means should be provided to allow the user to modify instructions already entered into the machine. This means that edit operations should be provided. Furthermore, it should be possible to regain control during execution without upsetting things to the extent that a complete retyping of a set of instructions is required to make a minor revision. Clearly a library facility to enable the user to save programs and data from session to session is essential.

The system should do some checking beyond that needed for translation when the user gives an instruction. The detection of contradictory or incomplete requests should initiate a series of questions by the system to obtain the information (as opposed to offering an error report and aborting). An example of a situation where a clarification request works well is the case when a variable which has not been given a value occurs in an expression.

An ideal system would have convenient facilities for graphical display at each terminal. In many instances a graph relates much more about what is going on than a table of numbers does.

The terminals should accept large quantities of pre-recorded data, say from punched cards or paper tape, or (slightly less desirable) there should be a central location to enter such data into a user's library.

The response time of the system is important. Excessively slow responses make the user nervous, and he feels that he is wasting his time. If the response is too quick, the user may feel pushed by the computer. An experimental interactive system is being designed at Berkeley to allow a user to specify a response time that is maintained within narrow limits. The user pays more for faster response.

It should be possible to initiate batch-processing jobs from a terminal (if the computing center has batch processing). This brings all the power of proven batch processing software within the reach of a remote user.

Admittedly these objectives are quite ambitious. Indeed, they may be too ambitious for a practical system. Quite a bit of costly hardware for terminals has been listed as desirable. This may make many terminals prohibitively expensive for all but the most affluent. Conceivably a small number of more sophisticated displays, etc. could be provided at central locations. As the number of users increases, the amount of library storage required can become astronomical and the cost of providing it prohibitive. Some features requiring rather sophisticated software have been advocated; such software is expensive to develop. The real pinch in designing interactive systems is in striking a satisfactory balance between compromises in services and the cost of development and operation.

Evaluation of interactive systems is very subjective because, aside from the question of whether operators perform their advertised operations, evaluation comes down to "how convenient is it?" "Does the convenience outweight the cost?", etc.

One scheme for evaluating a system is to make a limited release to users. User reactions, complaints, compliments are then weighed. This approach eliminates the bias of evaluation by the designer and is a real world evaluation. It may reveal serious weaknesses of the system which result from the way ideal objectives were compromised, or it may reveal that trivial modifications or extensions will give the user what he always wanted.

The ultimate criterion for evaluation is what the user thinks of the system.

5. SURVEY OF SOME INTERACTIVE SYSTEMS

NAPSS

The Numerical Analysis Problem Solving System is currently under development at Purdue University. A major part of the NAPSS system is a language close to natural mathematical rotation. The remainder of the system consists of software for the CDC 6500 computer which accepts the users problem and applies sound and reliable numerical analysis procedures.

One of the primary design goals is to make the system usable by users without training in either numerical analysis or computer programming. Thus, many rules and restrictions found in other computer languages are eliminated. Matrix, vector, and function manipulation are permitted with the same ease that languages such as FORTRAN permits manipulation of scalars and more operators are provided; for example , $\frac{d}{dx}$, , $^{-1}$, det.

Two types of assignment statements are permitted. The first is the usual one, A←E, where A is a variable (not necessarily a scalar) and E is an expression. If, for example, the value of E is a vector then A looses its previous identity and takes on the vector value of E. The second type of assignment statement is a symbolic assignment; A = E causes A to be set symbolically to the expression E. Unlike A←E, A = E does not result in the evaluation of E until a value of A is required.

```
A = 2 x X
X = 3 + Y
Y ← 1
B ← A
```

gives Y the value 1 and A the value 8. The order of the first three of these statements is immaterial.

An equation in NAPSS is two expressions separated by "=". Optionally statements may be labeled, for example EQ.23. The use of the label is equivalent to writing the equation.

Equations are used in SOLVE statements, which contain much of the power of the language. A SOLVE statement has the form

SOLVE EQS, FOR VARS (OPTIONS);

where EQS represents one or more equations or equation labels and VARS represents what is to be found. (OPTIONS) represents an optional list of information which may assist in solving the problem, specify the type or accuracy of answers desired, or direct the system to solve the equations in a particular way.

Conditional statements and iteration statements are very similar in format and operation to those in ALGOL. An extension in the iteration statement is FOR J,K $\Leftarrow$ 1 TO N DO.... ; which controls both J and K through a double nested loop. The ability to use the value of a quantity computed in a previous iteration of the loop is also provided--for example X [-3] is the value of X at the end of the third previous iteration.

Declaration statements are not required. In fact, the current version of the system does not use declaration statements at all--all declaration information is determined contextually.

The system is not yet available for use.

LINCOLN RECKONER

The Lincoln Reckoner is a time shared system designed for use in engineering and scientific research. It does not provide all of the facilities of a general purpose computer, but instead it consists of a library of elementary routines to perform numerical numerical computations on arrays. Thus Reckoner is basically a system designed for making use of routines, not writing them.

The library of routines available includes about 65 routines which perform the basic arithmetic operations on arrays of numbers, matrix arithmetic, array element shuffling, input, and output. These routines are used one at a time. Thus the user must parse his equations and translate them into a sequence of step by step operations. To use a routine, the user names it and its arguments. Clerical information such as array size is taken care of automatically.

The user can build his own procedures from the library routines and his other procedures. Such a procedure may have formal parameters, global or local variables.

The Reckoner system is currently running on Lincoln Laboratory's experimental computer, the TX-2. Basic components of the system include the time sharing system, the translator, and the library. The time sharing system performs three important services: collecting the input from the terminal and storing it in a buffer, keeping track of the name table, and maintaining the user's stack of maps which specify the contents of a current block of memory. After a full line of code is typed in, the translator parses the statement. If there are no errors, the translator forms a calling sequence and asks the time sharing system to push down the current memory map and stack on a new one for the subroutine that is to be called. When the subroutine is loaded into memory, it processes the calling sequence to find and classify its parameters, allocates temporary storage for local variables, and the proceeds to do its computations.

The system is very modular both in appearance to the user and in software implementation.

APL/360

APL/360 is a time sharing system developed at the Thomas J. Watson Research Center of IBM. It performs arithmetic operations on arrays of numbers. The system, which is currently in operation, is accessible to any user in the country via a dataphone link.

The APL/360 language appears very simple, but a full specification of the language was not obtained.

The most interesting feature of APL/360, and probably the one requiring the most software, is the work-space concept of program storage. Each user is able to define one or more workspaces in the public library, to put information into them, edit them, save them for future runs, and cause the programs entered into them to be executed.

A typical APL/360 run begins by dialing into the system on a dataphone and logging in. He then calls for an old workspace from a previous run or asks that a new workspace be defined. He can also communicate special information or request to the computer operator or another terminal. After his workspace is properly set-up he may call for execution, or if he desires he may employ incremental execution.

No figures were given to indicate the cost of using the system; however the long distance toll charges coupled with the relatively slow transmission rate probably makes this part of the expense prohibitive compared to other computer services.

AMTRAN (Automatic Mathematical TRANslation)

AMTRAN is an interactive system designed to permit the user to enter mathematical expressions in their natural format and receive immediate graphical and alphanumeric displays. Remote terminals consist of a large keyboard, two five inch storage oscilloscopes, and a special Selectric typewriter. They use voice-grade telephone lines. It is also possible to use simple typewriter terminals. The full scale terminal is estimated to cost from $5,000 to $10,000.

AMTRAN has three primary objectives:

1. A user with no background in computer technology should be able to solve relatively straight forward mathematical problems with little or no instruction in the AMTRAN system.

2. The more experienced user should be able to construct his own programs and operators at the keyboard to extend the range of the system and should have fast turnaround.

3. The system must be economically competitive with batch processing in speed and storage.

For the first goal AMTRAN provides operators like , $\frac{d}{dx}$, minimize, etc, automatic array arithmetic, automatic dimensioning of arrays, no declaration of variables, automatic working storage assignment, natural English I/O, picture formatting and other adjuncts to natural mathematics. Toward the second it has high level logical and branching operations, list processing and symbol manipulation capabilities and procedure and operator generation at a keyboard with facilities for using a disc library. Toward the third is has incremental compiler.

AMTRAN has both on-line conversational and batch-processing operation. The two modes can be combined. A "CALL THIS" operator is used to assign a name to and provide permanent storage for instructions. A library of commonly used instructions is available. Eventually rather powerful symbol manipulation facilities will be included - capable of manipulating strings (or arrays of strings) of alphanumeric and special characters plus special operators as , IF, GOTO, etc.

The terminal keyboard has numerous buttons (apparently of the order of 200) some have a permanent meaning and label. Others are available to represent operators or variables as designated by a user (who can then label them). The keyboard enters codes directly into the machine, bypassing the label decoding process. This makes implied multiplication possible from the keyboard (but not from typewriter). With its powerful buttons representing various available standard and user coded operators, and certain standard symbols, the keyboard provides rapid and accurate entry of mathematical statements. It also provides a means of controlling the displays on the two scopes (one of which is graphical; the other alphanumeric).

The current display system is strictly linear, but it is clear that conventional mathematical notation is feasible once actual display techniques are refined.

The typewriter has 88 characters including alphanumeric, special and mathematical symbols plus much of the Greek alphabet.

The 1966 version of AMTRAN was on a 1620, but bigger things are in the works.

It is conceivable that the language will never be completee for as time progresses more and more "goodies" can be included.

The article ignored the question of how the mathematical routines like those for , FIND ZERO, MINIMIZE, etc. work. When the article was written most of the effort seemed directed at language formulation and implementation.

KLERER-MAY (Two Dimensional Programming)

The basis of this system is a technique to accept mathematical equations and formulas as written down by the mathematician or scientist. Thus, this programming system allows subscripts and superscripts. This is where the name "Two-Dimensional Programming" arises.

This system is implemented in three interacting parts. These are the symbol recognizer, the translator and the compiler (FORTRAN). The first of these parts is treated very lightly in this article, but seems to be the most important, most difficult and most newly developed aspect of the system. This recognizer might use pattern recognition techniques. The approach most likely used is the geometric feature approach which is used in character recognition on other systems. This symbol recognizer recognizes Σ, $\int$, $\sqrt{\ }$, π, (,), [, and]; 16 Greek letters and all upper case and 12 lower case letters of the English alphabet. One nice feature is that it accepts and recognizes out-of-proportioned symbols. The translator is the usual language translator which takes this two-dimensional programming system and transforms it to a one-dimensional Fortran-like intermediate language. The FORTRAN compiler is then used.

The equipment required is a typical computer system coupled with a flexowriter. Thus, this system is no more expensive to install but gives an extra feature to programming.

This programming system has many of the Fortran statements plus one pertinent to this two-dimensional concept. An example program is given at the end.

Some important aspects of this system are; (1) a scanning of the instructions for translation into Fortran; (2) no dynamic storage allocation; but (3) semi-automatic dimensioning, i.e., deciding at run time the actual number of locations needed within the maximum number assigned during compilation; and (4) the system is an interactive system. This system also encounters the problem of ambiguities. The problem here is more complex since this system does not use delimiters. For example, how does the system interpret "SINACOSB" "$\frac{A}{2B}$", and "A/2B" The system handles the above as follows:

(i) SINACOSB: SINA*COSB instead of SIN(ACOSB)

(ii) $\frac{A}{2B}$: A/(2B) since the display division sign "_" is used

(iii) A/2B: (A/2)B since the Fortran division sign "/" is used.

But a fixed set of rules using the local context does not always give the correct interpretation. Thus immediately after the source program is read, the system's interpretation is listed on the high speed printer in a linear, Fortran-like intermediate language. An immediate response by the user to the system's interpretation resolves many of the ambiguities.

The advantages and disadvantages are related to the caliber of the user. The translator must be very sharp for an inexperienced user to write an efficient program. The only requirement on him is that he understand the I/O and declaration aspects of the system. However a very experienced person makes a lot of sacrifices in using the Klerer-May system. For instance, the translator probably uses only the obvious ways of doing the problem or expanding the two-dimensional equation into Fortran. Thus with the inefficiencies of the compiler this translator is likely to double the inefficiency of the code. Another major disadvantage of the system is that it is machine dependent.

KLERER - MAY EXAMPLE

Examples and explanations of some statements ....

I/O:

PRINT A - prints value of A in a floating point

PRINT γ⌈3⌉, X⌈4,2⌉ - prints γ as a 3-digit integer and X as a 7 place floating point number, 4 places to left of decimal point, 2 places to the right.

PRINT FORMAT# - to mix numeric with alphas

FORMAT # NUMBER IS XXX.XXX - prints "NUMBER IS" and the number by F7.3

READ i. - reads in a value for i

READ X_n,Y_n FROM n=1 to i.

DECLARATION:

MAXIMUM i = 500.

DIMENSION A = 100, B = 1000

{SUBROUTINE / PROCEDURE} (NAME)

CONDITIONALS:

If K = 0 K = 1 OTHERWISE ⌈ELSE⌉ PRINT A

Note: PARENTHESIS around the "IF" statement causes the statement immediately following to be executed whether it is true or not.

CULLER-FRIED SYSTEM

The Culler-Fried System is one name for two physically separate but direct descdendants of a system developed at Thompson Ramo Wooldridge Canoga Park, California, beginning in 1961. The first of these is the On-line Computer System (OLC) at the University of California at Santa Barbara. The OLC system is mainly used for research and teaching on the Santa Barbara campus. The second expanded version of the original system, has been implemented at TRW Systems in Redondo Beach, California and has been operating since late 1964. Both of these systems use small, relatively slow machines that are dedicated to the system. A larger,faster one is in progress.

These systems are primarily oriented toward mathematical analysis and permit significant numerical calculations. The system does not require completion of a program before beginning execution, and permits a continuous interchange of information between the user and the machine. The system allows the user to work directly with "larger" mathematical entities than scalars, such as functions, or vectors.

The OLC version uses a RW-400(AN/FSQ-27) computer with an add time of 50 micro-sec (and a 1024 word core). The I/O facilities are a 5" storage oscilloscope, 2 keyboards with 48 keys at each terminal, and there is no hard copy available. The input keyboard is an array of push buttons. The system environment is a stand alone system with a 80K word drum and a magnetic tape available for storage of data and console programs.

The TRW version uses a Bunker-Ramo 340 with an add time of 12 micro-sec and a 16 K core. The I/O facilities are a 5" storage oscilloscope, 2 keyboards each with 48 keys at each terminal, a calcomp plotter, and an output typewriter shared among 4 terminals. The system environment is 4 simultaneous users, console programs stored on a drum, and magnetic tape available for storage of data and console programs.

Scalars, vectors or complex vectors may be denoted by a letter A-Z used in conjunction with operator levels I, II, or III respectively.

The use of operator levels separates scalar and vector operations. All operations on "Level II" modify a vector that has been loaded into a "function register." This "register" is a group of up to 124 memory locations used for temporary storage of the elements of a vector, while operations upon it are being performed. Four of these function registers, the X, Y, U, and V, registers, are used by the system, and the user has available a large number of operators which affect their contents. Users can define their own operators by defining a sequence of system operators.

Some of the other good features of the system are the availability of logical operations and branching in console programs, the ability to inspect console programs and to modify them in a general way, and convenient storage for user data and programs between console sessions.

Some bad features are the inability to substitute argument variables for dummy variables in console programs, the inability to define local variables in a console program and the necessity for the user to use polish notations in all computations and to keep track of memory storage.

POSE

POSE, a language for posing problems to a computer was designed to simplify the task of solving a broad class of numerical problems. The method of solution is automatically provided without requiring the user to have a knowledge of the numerical methods or of the logic involved. Certain clerical operations required in Fortran programming are handled automatically by POSE--for example, defining common storage areas, writing format statements, and development of control logic. POSE also provides for a variety of alternate display procedures such as two-dimensional graphs and perspective views of three-dimensional graphs.

POSE is presently being implemented for the IBM System/360 and IBM 1800. POSE contains as subsets both FORTRAN IV and assembly language. The special capabilities to be built into POSE enable it automatically to solve the following types of problems:

- ordinary differential equations (I.V. Prob)
- evaluation of multiple integrals
- transcendental algebraic equations
- systems of linear equations
- matrix arithmetic and inversion
- table lookup and N th order interpolation
- basic statistical computations
- function evaluation with automatic parameter variation

The description of the language is limited primarily to four annotated sample programs. At that time no implementation of the language existed, so the output which these programs should produce was not given. A POSE program is divided into program segments, with the beginning of a program segment indicated by a label of the form S. {<digit>}... The first statement in a program segment identifies the type of calculation to be performed--for example, FUNCTION evaluation, MATRIX arithmetic, INTEGRAL evaluation, DIFF. EQ. solution, etc. A special program segment identified as the CALCULATION SEQUENCE is used to control the execution of other segments in the program. If this segment is not included, then execution begins with the first statement of the first segment after reading the first data set. In this case, execution of the statement "EVALUATION END" indicates that the process should be repeated for the next data set. Typically the statements within a segment supply parameters controlling the calculation of the segment rather than explicitly specifying the actual order of calculations.

Sample Program

Evaluate the function $G(x,y) = \sin x^y + \exp(0.053\ xy) + F(x)$, where F(x) is supplied in tabular form. The values for x are to range from 0 to 5 in steps of 0.1 and the values for y are to range from 2 to 3 in steps of 0.25. The values of G(x,y) are to be printed with corresponding values of x and y under the column heading

```
        X             Y            WEIGHT
       FEET          FEET
```

A perspective plot of G versus x and y as viewed from the point (10,10,10) is also desired. In the program below, "RPT.10" is a reference to a standard output report format, and the Z used in the data is a dummy variable.

```
S.O    CALCULATION SEQUENCE
       READ DATA
       EXECUTE S.10
       PROBLEM END

S.10   PLOT PERSPECTIVE(G.VS.X,Y) FROM (10.,10.,10.)
       RANGE OF X = 0.(.1)5.
       RANGE OF Y = 2.(.25)3.
       G = SIN(X**Y) + EXP(.053*X*Y) + TABLU(F(X),2)
       PRINT RPT.10 (X,Y,G) TITLE(X:FEET,Y:FEET,WEIGHT)

DATA   CASE 1

TABLE F(Z), 6 ENTRIES

Z(1) = 0., F(1)=1.,    Z(2)= 2., F(2)=1.25
Z(3)=3. , F(3)=1.5,    Z(4)=3.35, F(4)=1.596
Z(5)=4.2, F(5)=1.65943 , Z(6)=5.5,  F(6)=2.0

END
```

The implementation of POSE on the IBM System/360 and IBM 1800 was expected to require less than one man-year of effort for each computer. Two factors contributed to this (perhaps optimistic) prediction. First, the POSE compiler was being designed to produce a FORTRAN IV program as its output, and the source statements were already Fortran-like in many respects (it was not stated how assembly language statements would be handled). Secondly, POSE was an extension of a previous system, Engineering Analysis and Simulation Language (EASL), which was already operational on the IBM 7094, and thus the implementation of POSE involved "only" an augmented reprogramming of an existing system.

ICES - Integrated Civil Engineering System

ICES was developed under Daniel Ross in the Civil Engineering Department of MIT. The design philosophy was based upon providing accurate, immediate problem analysis to aid in decision making. The designers also wished to relieve the engineer of the clerical tedium associated with storage allocation, I/O devices, and I/O formats.

ICES provides two types of languages - a procedure oriented language called ICETRAN and a number of problem oriented languages. ICETRAN is an extension of FORTRAN which allows use of the data management and dynamic allocation facilities in the system. ICETRAN also has provisions for list processing and matrix manipulation. ICETRAN is input to a "precompiler" which produces a FORTRAN program for the FORTRAN compiler.

ICES integrates a number of subsystems - for example, in solving a single problem, an engineer may need access to procedure concerned with soil mechanics as well as structural design. Each subsystem has its own problem oriented interactive language. The following examples are from a present subsystem, COGO (Coordinate Geometry), which was developed at MIT before ICES.

The command:

```
STORE POINT 10 X 100.5 Y  960
```

defines a point labeled 960 with the given coordinates. The statement

```
DISTANCE 2  3
```

provides the user with the distance between points 2 and 3.

A distinguishing feature of ICES is its self-modifying capabilities. ICES provides a "command definition language," a problem oriented language for producing commands of a problem oriented language. This allow the user to generate his own commands in terms of existing commands.

The author claims that the system is machine independent. The original version was implemented on a 65K 360/40 with several disc drives and a number of graphical I/O devices.

MAP (Mathematical Analysis Program)

MAP was written at MIT and is part of their time sharing system. It is intended for those people more interested in learning mathematical procedures than with learning programming or obtaining meaningful results. Problems have to be broken down into their basic procedures in order to use MAP.

MAP gives a great deal of interaction between user and system, perhaps too much, but this is in line with the objective of emphasis on procedures rather than results. The user is allowed to make decisions and insert changes at any time during the course of execution. There are no logical variables so all decisions must be made by the user and explicitly entered into the system through the keyboard. Besides promoting interaction this is also necessary since the time sharing system does not service a terminal often unless the user interacts often. The system uses normal English conversation and is fairly easy to understand; the user is restricted to shorter commands to minimize his typing and to ease the systems programmer's burden.

The MAP system calls on subsystems to carry out particular functions, e.g., integration, convolutions, Fourier transforms, etc. The designers theorized that 30-50 such subsystems would be desirable but in 1966 only 14 were implemented. However, the system appeared to be reasonably satisfactory with 14. The user may also use other languages or call on subroutines in other languages (linked through MAD). No local variables are allowed. This and the need for setting up linkage, assumes that the user is reasonably familiar with the subroutine being called upon.

Data can be described in a variety of formats (both for input and output), but input formulas must be linear due to the conventional keyboard used. Functions can be specified in tables (equally spaced or otherwise) or as explicit formulas, e.g., y(x) = sint(x) **2-a. Sint is used rather than 'sin' to distinguigh the built-in function from one which is user defined. It is understood that y is the dependent variable, x the independent variable and "a" is a constant. The system asks the user to stipulate the value of 'a', the range of x and an interval width for discrete x. One disadvantage here is that only one independent variable is allowed, however, the user is free to change the values of constants. Results are not displayed unless specifically requested. The system's extensive editing features can spot some typing errors and ask for clarification. It also detects undefined variables or constants.

The system has the capability of providing CRT plots of data in a variety of formats though this feature is rather expensive to provide at each console. The user is also permitted to define and name procedures and save them.

MAP provides error messages in cases where operations might not be meaningful. Most procedures are carried out with accuracy equivalent to "three-point fitting". If this is insufficient the user must supply his own algorithms. Some procedures provide error estimates but as most users of this system are numerically naive these might not carry much weight.

This system, like other interactive systems, is inefficient for carrying out long computations, however, a remote batch mode will soon be provided.

6. AUTOMATED NUMERICAL ANALYSIS

It has long been desirable to have a computer system to permit problems to be stated in languages appropriate to the problem and to provide for their solution without the services of specially trained programmers and analysts. The basic idea in numerical analysis is to make the computer behave as if it had some of the knowledge, ability and insight of a professional numerical analyst. Such a system (which usually consists of a compiler and/or an interpreter and a set of numerical analysis procedures) provides automated numerical analysis.

Most attempts at designing an automated numerical analysis system have centered on three important points:

1. A de-emphasis of the general-purpose property of a language and an increased emphasis on the problem-oriented aspects.
2. Establishment of a problem-oriented conversational environment.
3. Achievement of machine independence both internally and externally.

Most of the design objectives that follow are implemented by one or more of the following systems, NAPSS, RECKONER, KLERER-MAY, AMTRAN and MAP.

Design Objectives;

- machine independence
- non-linear notation
- easy input language, self-explanatory, tailored input language for mathematical users
- the flexibility of a procedural language
    - a. local and global variables
    - b. procedure parameters
    - c. logical operators
- ability to work with larger entities than scalars, such as functions, vectors and arrays
    - a. automatic array arithmetic
- solve operators which are usually implemented by a polyalgorithm approach
- conversational mode
- access to other programming languages
- elimination of almost all clerical operations

a. virtual memory
b. automatic dimensioning
c. automatic type and mode

reduce turn around time

To accomplish these objectives many techniques have been used. Many of the objectives can be met thru software design but some require special hardware. For example the Klerer-May system uses a flexowriter to achieve non-linear notation. To allow interaction between the machine and the user during execution incremental execution is used. Interpretive execution allows one to eliminate declarative statements, entry of undefined variables at execution time and editing. To use "solve" operators and reduce the effort of the user and reduce the elapse time, a polyalgorithm approach was designed.

The main objectives of the polyalgorithm approach are the following: a. reliability, b. error control, c. efficiency, d. flexibility, e. common sense, and f. simplicity of use. Some of the objectives are in direct opposition and one does not know how to meet some of the objectives for certain numerical analysis problems. A short discussion of the problems and compromises of the implementation of common sense, error control, flexibility versus simplicity of use, and reliability versus efficiency is given.

A. Common Sense is the application of one or more of a large number of simple rules to an appropriate part of a large body of information. The main problems encountered in attempting to implement common sense are the large number of logical decisions, and the collection of a large body of information.

B. Error Control - The problem here is not only to control the error in each algorithm, but to also control the overall error in the program. Overall reliability is not great and the implementors of such an approach are the first to mention this fact. To achieve reasonable error control for single algorithms, they imitate the professional numerical analysts. They start with some sound numerical analysis algorithm and use a series of a postiori tests. Overall error control is left up to the user as always.

C. Flexibility vs. Simplicity of Use - Here one wants to allow simple statements such as "integrate this function", and "solve this equation", but also to provide flexibility to experienced numerical analysts. To achieve this one can use optional qualifying phrases in statements. This approach preserves simplicity in a flexible environment. It is also concise and natural to the user.

D. Reliability vs. Efficiency - The main question is: How does one prevent a polyalgorithm from going through all the computation it does for a difficult problem when we have an elementary problem which any good method solves immediately? One aid to efficiency is additional information, either internal or external. The internal method is to try a simple method first and work up from there allowing the polyalgorithm to learn more about the problem. The external information can come from the user prior to execution, the user during execution and from other polyalgorithms.

The design of an automated numerical analysis system encounters many problems and requires many compromises. The basic compromise of this type of system seems to be that it requires a little bit more of the user than originally intended.

6.1 Comparison of 5 Automatic Numerical Analysis Systems

	AMTRAN	KLERER -MAY	NAPSS	RECKONER	MAP
Nonlinear Language	No	Yes	No	No	No
Self-documenting Input	Yes	Yes	Yes	Yes	Yes
Functions	Yes	No	Yes	Yes	Yes
Equations	Yes	No	Yes	No	No
Automatic Array Arith:	Yes	No	Yes	Yes	No
Complex Arith	Yes	No	Yes	No	No
High Level Operators	Yes	Yes	Yes	No	No
Execution Request for Undefined Variables	No	Yes	Yes	No	Yes
Solve Operators	No	No	Yes	No	No
Local Variables	No	No	Yes	Yes	No
Global Variables	Yes	Yes	Yes	Yes	Yes
Incremental Execution	Yes	Yes	Yes	Yes	Yes
Access to Other Procedural Lang.	No	No	No	No	Yes(MAD)

6.2 Automatic Differential Equation Solvers

A good automatic differential equation solver should have the following properties:

(1) ability to pick 'good' starting values (for multistep methods)

(2) numerical stability

(3) automatic error monitoring

(4) relative ease of use

(5) efficiency (in time and memory requirements).

Error monitoring includes the ability to use information about the error when necessary to switch to a better method, change the integration interval or inform the user.

There is a variety of systems which have some or all of these features. The first example is DEMON. It consists of a COBOL program which takes a system of differential equations, a stepsize and intial values and generates a FORTRAN program. This system has neither the flexibility of providing initial values and stepsizes nor the ability to monitor error. The latter could be significant since the method used is 2nd order Runge-Kutta.

One system using a finite Taylor's series expansion was proposed in 1960 by Gibbons. All variables are represented in terms of coefficients of such an expansion. The initial values together with the differential operator provide values for the constant terms in the expansions for the function and its derivatives. By integration and repetition of this procedure the remaining coefficients can be generated recursively.

A similar system is AUTOMAST (Automatic Mathematical Analysis and Symbolic Translation) implemented at Washington University. This system in addition to Taylor's series manipulation can, as the name implies, provide symbolic analysis, i.e., parameters can be carried symbolically into the solution. This procedure substitutes the given expressions into a Taylor expansion and uses initial values and integrations to solve for the unknown coefficients. All manipulations are carried out symbolically. The input for this system consists of a system of ordinary differential equations, initial conditions, and identification of dependent and independent variables. The output is a Taylor's series expansion for each dependent variable with an accompanying error estimate. This system might be extended to be interactive and to be able to optimize the number of terms in the expansion in terms of truncation and round-off error and requirements for time and memory. At present, however, it is somewhat deficient in handling error and it is up to the user to provide the degree of approximation. In addition the system has the disadvantage that this method is invalid for problems having nonanalytic solutions.

In 1962 Nordsieck proposed a method for automatic integration of differential equations. This was motivated by a desire to stabilize the Taylor's series approach and to incorporate automatic choice of starting values and initial and subsequent interval size. This method is intended to minimize the amount of computation for a given accuracy and can be applied to almost any system of equations. This method was incorporated in a system at the University of Illinois on Illiac II. However, it did not perform well so a completely new system was developed. This new system is capable of handling systems of any order differential equations, and uses a predictor-corrector. The user can stipulate the degree of approximation, coefficients of the method, number of corrector iterations and the step size.

In addition the system contains automatic procedures for generating any of these. The system is interactive and printer plots can be generated. This system provides most of the desired features for automatic integration.

The final system considered runs on the 7094 at Princeton University. It is written entirely in FORTRAN except for two MAP routines used in producing graphs. It uses Adams predictor corrector formulas (of orders 3,4,5,6) and incorporates automatic choice of starting values and internal control logic for bounding the error. The user can also specify these and provide his own formulas. The system is designed to run in segments, i.e., during execution it can be suspended and later resumed without starting over at the beginning. This has merit in multiprogramming systems or in cases where the problem is too large to be run in one job or if intermediate results are desired. The system appears to be highly FORTRAN oriented which may limit its appeal.

6.3 <u>OMNITAB</u> – A general purpose non-interactive system. OMNITAB was written for use at the National Bureau of Standards. It seems to be designed for those people who don't know anything about computers and don't want to bother to learn. Its format gives the appearance of a desk calculator whose storage is considered as a 101 row by 46 column work sheet. The system is composed of numerical analysis and statistical routines, tables of special functions, matrix operations and the usual arithmetic operations. The instruction set consists of simple English words, e.g., ADD, EXCHANGE, READ, so it is fairly easy to use. The input is in free format and is especially convenient for tabular data. The instruction format is also relatively flexible (the system only considers the first six characters in decoding the operation). The operands (generally column numbers) must be in a certain order but this usually is the natural one. Any other words than the first on a card are ignored so they can be made as conversational as desired. OMNITAB operates under the IBSYS system on the 7094. Some of the particular features provided are differences, least squares fitting, Bessel functions, matrix operations (including inversion and finding eigenvalues) and integration and differentiation (using the LaGrange interpolating polynomial). The normal mode is sequentially interpretive with instructions only being executed once. In addition there is a REPEAT instruction which provides some looping capabilities. This language has a very good manual, a wide variety of built-in routines and is fairly easy to use as no prior programming knowledge is necessary. However, due to its simplicity it does not provide many of the desirable facilities of more popular programming languages, e.g., named variables or procedures.

6.4 Automated Statistical Systems. There are many different statistical packages available as generally every statistical library has their own set of routines. Some examples of university produced systems are BMD (UCLA), MEDCOMP (University of Cincinnati) and P-STAT (Princeton). Some of the desirable features of a statistical routine are: (1) free format for input, (2) a good description of just what each routine does, and (3) easy to learn instruction or control card format.

The first system has few of these properties. This is COSMOS (Courtauld's own system for matrix operations and statistics) which does provide many of the basic statistical procedures. However, its input formats are different for matrices, constants and statistical tables, and the control cards describing operations are rather complicated so this system is not likely to have a very wide distribution.

Another system is MSA (Multivariate Statistical Analysis) which has some of the same disadvantages as COSMOS. It consists of a group of segments each of which carries out some statistical procedure, e.g., factor analysis, maximum likelihood, regression, correlation and covariance analysis. The user's control cards specify which segment is desired and though these are relatively simple the data must be formatted and the descriptions for this can be very complicated.

A better system is STAT-PAK (Berkeley) which is written in a FORTRAN II subset. It is compatible from one machine to another. It is written for a system with card input, printer output and at least an 8K core memory. The input consists of header cards describing the procedure desired, followed by a parameter card specifying any variables needed by this particular routine and the remainder is data. This system handles regression analysis, estimation, analysis of variance, tests of hypotheses, matrix manipulations, etc. There is an operation manual for the entire package plus individual write-ups and flow charts. This system provides many of the desirable features of a statistical package and though it is FORTRAN oriented the user need not be very familiar with FORTRAN.

REFERENCES

A. REFERENCE BOOK ON NUMERICAL ANALYSIS

1. Computer Approximations, J.F. Hart et al, John Wiley, 1968.
2. Handbook of Mathematical Functions, National Bureau of Standards, AMS #55, 1964.
3. Handbook for Computing Elementary Functions, L.A. Lyusternik et al, Pergamon Press, 1965 (translation).
4. Mathematical Methods for Digital Computers, A.Ralston and H.S. Wilf, Vol. 1 (1960) and Vol. 2 (1967), John Wiley
5. Springer Handbook on Automatic Computation (parts appear also as articles in Numerische Mathematik)

B. GENERAL NUMERICAL ANALYSIS TEXTS

6. Elementary Numerical Analysis, S.D. Conte
7. Elements of Numerical Analysis, P. Henrici
8. Introduction to Numerical Analysis, F.B. Hildebrand
9. Analysis of Numerical Methods, E. Isaacson and H.B. Keller
10. First Course in Numerical Analysis, A. Ralston

C. SPECIALIZED NUMERICAL ANALYSIS TEXTS

11. Numerical Integration, P. Davis and P. Rabinowitz
12. Computational Methods of Linear Algebra, D.K. Faddeev and V. N. Faddeeva
13. Finite Difference Methods for Partial Differential Equations G. Forsythe and W. Wasow
14. Discrete Variable Methods in Ordinary Differential Equations P. Henrici
15. Matrix Iterative Analysis, R. Varga
16. Iterative Methods for the Solution of Equations, J.F. Traub.
17. The Algebraic Eigenvalue Problem, J. Wilkinson

D. AUTOMATED NUMERICAL ANALYSIS AND INTERACTIVE SYSTEMS

18. Proceedings ACM Symposium for Experimental Applied Mathematics, edited by M. Klerer and J. Reinfelds, 1968. This book contains discussions of almost all interactive systems under development in 1967 and also has a number of other papers pertinent to this area. Papers which appear there are not listed individually.
19. The AMTRAM system, Wood, Reinfelds, Seitz and Clery, Datamation, Oct., 1966.
20. An Integrated computer system for engineering problem solving, (ICES). D. Roos, Proc. FJCC, 1965, pp 1-11.
21. JOSS: A designer's view of an experimental on-line computing system. J.C. Shaw, Proc. FJCC, 1964, pp 455-464.

22. The Lincoln Recokoner, etc. Proc. FJCC, 1966, pp 433-444.
23. MAP: Man-machine communication in on-line math.
24. NAPSS: A numerical analysis problem solving system, J. Rice and S. Rosen, Proc. ACM 1966, pp 51-58.
25. OMINTAB, NBS Handbook #101, 1966, J. Hilsenrath et al.
26. POSE: A Language for posing problems to a computer, S. Schlesinger and L. Sashkin, Comm. ACM 10 (1967) 279-285.
27. The status of systems for on-line mathematical assistance, A. Ruyle, J. Brackett and R. Kaplow, Proc. ACM, 1967, pp 151-167.
28. Two-dimensional programming, M. Klerer and May, Proc. FJCC 1965, pp 63-75.
29. An appraisal of least squares programs for the electronic computer from the users point of view, J.W. Longley, J. Amer. Stat. Assoc. 62(1967) 819-841.
30. Proposal for subroutine comparison, B. Witte, SICNUM Newsletter, 1967.
31. Numerical solution of ordinary differential equations at a remote terminal, W. Gear, Proc. ACM, 1966.
32. A program for the automatic integration of differential equations, Computer J., (1960) pp 108-111.
33. AUTOMAST: Ball and Berns, Comm. ACM, 9(1966) pp 626-633.
34. STAT-PAK, Henschke and Shannon, Comm. ACM, 10(1967).
35. The Multivariate Statistical Analyser, Jone, Harvard University Press, 1964.