REPORT RESUMES

ED 011 862 SE 001 762

PROCEEDINGS ON THE CONFERENCE ON THE IMPACT OF COMPUTERS ON EDUCATION IN ENGINEERING DESIGN (CHICAGO, APRIL 21-23,1966).

BY- FENVES, STEVEN J. AND OTHERS

COMMISSION ON ENGINEERING EDUC., WASHINGTON, D.C.

PUB DATE 66

EDRS PRICE MF-$0.27 HC-$7.20 180P.

DESCRIPTORS- *CONFERENCE REPORTS, *CURRICULUM DEVELOPMENT, *ENGINEERING EDUCATION, *TECHNICAL EDUCATION, COMPUTER ASSISTED INSTRUCTION, EDUCATIONAL TECHNOLOGY, INSTRUCTION, INFORMATION RETRIEVAL, SCIENCE EDUCATION, NATIONAL SCIENCE FOUNDATION, CHICAGO, DISTRICT OF COLUMBIA

PAPERS PRESENTED AT A 1966 CONFERENCE ON ENGINEERING EDUCATION ARE INCLUDED IN THIS CONFERENCE REPORT. THE CONFERENCE WAS PRIMARILY CONCERNED WITH THE IMPACT OF COMPUTERS ON EDUCATION IN ENGINEERING DESIGN. PARTICIPANTS WERE FACULTY MEMBERS FROM SCHOOLS OF ENGINEERING, WHO HAVE THE RESPONSIBILITY FOR COURSE AND CURRICULUM DESIGN. A LONG-RANGE OBJECTIVE WAS TO SET DIRECTIONS FOR ENGINEERING EDUCATION AND RESEARCH IN DESIGN AND ANALYSIS, WHEREVER THE LATTER WAS EXPECTED TO CHANGE UNDER THE IMPACT OF COMPUTER CONCEPTS. SESSIONS OF THE CONFERENCE WERE DEVOTED TO (1) APPLICATIONS IN ENGINEERING DESIGN, (2) COMPUTERS IN ENGINEERING DESIGN AT UNIVERSITIES, (3) A MANUFACTURERS FORUM, AND (4) FUTURE PROSPECTS AND REQUIREMENTS. SUMMARIES OF DISCUSSIONS THAT FOLLOWED PRESENTATIONS OF PAPERS ARE INCLUDED. (AG)

THE COMMISSION ON ENGINEERING EDUCATION

1501 NEW HAMPSHIRE AVENUE N.W. WASHINGTON D.C. 20036

PROCEEDINGS OF THE CONFERENCE ON THE IMPACT OF COMPUTERS ON EDUCATION IN ENGINEERING DESIGN

THE COMMISSION ON ENGINEERING EDUCATION
1501 NEW HAMPSHIRE AVENUE, N. W., WASHINGTON, D. C. 20036

Program and Table of Contents

Section A

Section B

Section C

Section D

Section E

Section F

Section G

PROCEEDINGS
OF THE
CONFERENCE ON THE IMPACT OF COMPUTERS ON EDUCATION IN ENGINEERING DESIGN

FOREWORD

The Commission on Engineering Education, through its Advisory Committee on The Use of Computers and Mathematical Techniques in Engineering Design, sponsored a conference at the University of Illinois at Chicago Circle, Chicago, Illinois, April 21-23, 1966, dealing with the impact of computers on education in engineering design. The conference steering committee was composed of:

PROFESSOR STEVEN J. FENVES, *Conference Chairman*
Dept. of Civil Engineering
University of Illinois

DR. FRANKLIN H. BRANIN, JR.
Data Systems Division
IBM Corporation

DR. SULLIVAN A. CAMPBELL
Xerox Corporation

PROFESSOR BRICE CARNAHAN
Dept. of Chemical and Metallurgical Engineering
University of Michigan

PROFESSOR SAMUEL E. SHAPIRO
Department of Systems Engineering
University of Illinois at Congress Circle

The conference was intended to serve the faculty members of engineering schools who are responsible for the development of courses and curricula in engineering design. Its immediate objective was to explore all the facets that interact on design, design education, and computer technology. The long-range objective was to set directions for engineering education and research both in design and analysis, wherever the latter is expected to change under the impact of computer concepts.

These proceedings comprise most of the papers presented by the participants, as well as the discussions which followed in some cases. The conference program and a list of those who attended it are included.

The conference was partially financed by a grant from the National Science Foundation to the Commission on Engineering Education.

OPENING SESSION

SECTION A

CHAIRMAN: S. J. FENVES, *Prof. of Civil Engineering*
University of Illinois

THE IMPACT OF COMPUTERS ON EDUCATION IN ENGINEERING DESIGN

J. C. R. LICKLIDER
Thomas J. Watson Research Center
International Business Machines Corporation
Yorktown, N. Y.

(Summary)

The keynote is the rate of advance of the technology that provides the medium and the means—and constitutes in large part the context and the object—of engineering design. It is a high note. Several of the basic capabilities of informational technology, such as capacity for storage and speed of processing, have been doubling every two years. Over all, though undoubtedly not in every sector of the technology, a high rate of change seems likely to persist throughout the university years of engineers who are now freshmen and well on into their postgraduate careers. Engineering education must do its best to prepare them to expect, to master, to employ, and, indeed, to contribute to the development of the more effective ways of thinking and working that the advanced technology will make possible.

At the present time, we see a major wave of technical advance about to break upon the field of engineering design. It may be only the first of a succession of waves, or it may be the great main force that will shape the coming generation. All that is sure is that the wave is massive and that its impact will be great.

The essential nature of the cresting wave is caught up by the current phrase, "close interaction."

The process of engineering design involves interplay between creative and routine activities, between activities that are heuristic, such as formulation of hypotheses, selection of approaches, assessment of values, and activities that are algorithmic, such as determination of costs and calculation of stresses. In design by classical methods, those two kinds of activity were insofar as possible separated from each other, the designer attending to the heuristic part himself and assigning to draftsmen, clerks, and computers the part that he could lay out step-by-step in explicitly detailed procedure. That approach was successful in inverse proportion to the challenge of the problem, for only when the design was confined to preformed channels of solution could anything like a clean separation of the algorithmic from the heuristic be achieved. It became obvious, indeed, that in truly creative engineering design the two kinds of activity guide and support each other in intimate interplay and that, if there is such a thing as a phase of creative inspiration followed by a phase of clerical execution, the total effort involves hundreds of thousands of such phases.

Thus creative engineering design requires close interaction between the heuristic factor and the algorithmic factor. Because people are so greatly superior to machines in respect of the former and machines are so greatly superior to people in respect of the latter, the requirement is essentially a requirement for close interaction between men and computers. And that is precisely what is afforded by the new informational technology, which makes it possible:

1. To bring the time scale of man-computer interaction down to seconds, or even milliseconds, and thus to substitute for the old, coarse concatenation of human operations and machine operations a fine, tight interweaving in which even the smallest area contains both the man's heuristic and the machine's algorithmic threads.
2. To rescale the computer's contribution from uninterrupted execution of an ad hoc monolithic program prepared to solve an entire problem or carry out an overall task to responsive execution of general purpose atomic programs prepared to place a broad repertoire of fundamental information operations at the designer's fingertips.
3. To free the designer from the dilemma of having either to plunge forward one creative step to another with only the support of his own engineering estimates and artistic intuitions or to wait after each creative step for slow and laborious implementation and testing.

The close-interaction approach to engineering design depends upon synergic use of new hardware, new software, and new techniques. In this conference, we shall examine all three.

In the field of hardware, the main areas are storage, processing, transmission, and input-output. In storage, the most dramatic advance is the advance up the scale of capacity. Stores capable of holding a million analog images or a trillion binary digits and affording access in a few seconds to any image or any bit are in advanced stages of development.

The trillion-bit digital stores seem especially significant to me because they will make it possible to maintain vast libraries of engineering information in readiness for rapid access and for processing under program control.

In processing, the key advance is multiple access through time-sharing and multiprocessing, with memory protection, dynamic relocation, and related features that make it possible for a team or community of engineers to use extensive (and expensive) facilities in a coordinated and efficient way. In transmission, the main relevant movement is the growth of digital transmission services and the melding of digital transmission with digital computation—which will let the user work in or near his office with information stored at diverse remote locations and processed wherever the computers are. In input-output, the critical area of advance is that of consoles for on-line man-computer interaction. Most of the basic capabilities are provided by one or another of the new-generation consoles. As consoles move from their positions of relative obscurity in traditional computing systems to become the main foci of interaction between the users and the facilities of on-line information systems, there is intensive development aimed at improving the basic functions of display and control, integrating them more effectively for convenience and rapidity of interaction, and making them widely available and affordable.

The software requirements imposed by engineering design and related multiple-access, interactive applications are more basic and more difficult, in my assessment, than are the corresponding hardware requirements. The general requirement is the "software base," the on-line library of widely useful subprograms; the on-line languages and program-processing programs through which users will call and apply the subprogram from the library; the on-line programming systems that will facilitate preparation, testing, modification, documentation, and communication of new programs and subprograms, and on-line, controlled-access data banks. Problems raised by those general software requirements are being attacked vigorously now, and the most basic requirements are gradually being met by research and development projects in universities, the automobile and aerospace industries, the computer and communication industries, and several governmental and government-related non-profit organizations. Only two years ago, a large fraction of the progress in this area could be subsumed by citing "Sketchpad" and other programs at Massachusetts Institute of Technology in Cambridge and at the Lincoln Laboratory, the DAC-I Project at General Motors, and related work at Bolt Beranek and Newman, System Development Corporation, and very few other other places. In the interim, the field has burgeoned.

Under "new methods and techniques" in the phrase "new hardware, new software, and new methods and techniques," I meant to include new ways of using on-line facilities for engineering design. It is now well understood that the main promise of the on-line approach is not merely to automate drafting, but no one I know thinks he yet sees clearly just how designers will use the new facilities or just what the process of design will be within the new context. The new ways of working and thinking will evolve out of experimental and pilot-operational use. One of the main responsibilities of educators, therefore, is to incorporate such use of on-line facilities into the process of education in engineering design. Students should learn the new methods and techniques not so much by being taught them, as by contributing to their conception and development.

What beyond on-line interactive design? Perhaps it is not too soon to think about that. My own best guess is that, when many engineers are working within the context of an on-line community, there will be a marked acceleration in the understanding of the heuristic aspects of design. More and more of the exploration of partly random, partly controlled variations on themes, for example, will be carried out by computer programs, and designers will tend increasingly to become the strategists, rather than the conductors, of explorations and experiments. Designers will function also as evaluators, of course, but perhaps even in that area they will become strategists—students of the processes through which "time" performs its ultimate evaluations. In any event, new possibilities and new opportunities will arise, and (from the somewhat non-central but nonetheless interesting point of view from which I see it) the future of engineering design looks both challenging and exciting.

SOME COMMENTS ON THE FUTURE PRACTICE OF ENGINEERING

C. L. Miller
Massachusetts Institute of Technology
Cambridge, Mass.

THE YOUNG ENGINEER—1976

The young engineers who will be 28 years old in 1976 will be entering college as freshmen this fall. While 1976 sounds like a long way off to you and me, age 28 sounds quite young to all of us. Certainly we will want our 28-year-old engineers to be well prepared for their professional work. This means that the formal education we give these entering freshmen during the next four to eight years should reflect engineering as it will be practiced in 1976, that far-off date.

THE NEW TECHNOLOGY—1976

The new technology for engineering which is already available has been excellently described in the previous address by Dr. Licklider. With our concern for the 28-year-old engineer in 1976, we must project this new technology ahead for at least a decade of what is sure to be a period of accelerated progress.

While no one can predict what the technology for engineering will be in 1976, it will surely be much more advanced than that available for exploitation today. One conservative approach to projecting ahead to 1976 is to say that the most advanced of the new technology which is just emerging in research environments in 1966 will be commonplace in design and production environments in 1976.

THE PAST DECADE—ELECTRONIC CALCULATION

The past decade might be characterized as the decade of electronic calculation in engineering. Since 1956, excellent progress has been made in the application of the computer for engineering calculations and in the use of the computer as an aid in engineering analysis. In engineering schools the focus has been on the teaching of elementary programming and an introduction to appropriate numerical methods. In addition, extensive use has been made of the computer in engineering schools for research calculations. During the past decade, interest in the potential role of the computer in engineering design has developed, and some progress has been made. Excellent examples of such progress will be presented during the balance of this program.

THE NEXT DECADE—INFORMATION SYSTEMS

As we look ahead to the next decade, there is ample evidence that it will be characterized as the decade of engineering information systems. What has been done in the past for engineering calculations will be done in the future for the whole area of engineering communications, decision-making, and systems design. We will be concerned with the computer as a component in large-scale engineering information systems which will encompass the total spectrum of information acquisition, processing, storage, retrieval, and communication. We will be concerned not only with computer technology, but also with information technology in the broad sense of the word.

THE NEW CAPABILITY—A NEW PRACTICE

Information technology and the information systems of 1976 will represent new kinds of capabilities for engineering practice, not just improvements on the old capabilities of past decades. To exploit the new capabilities will not be as simple as adding them to the old capabilities. It may take a rather complete restructuring of the practice of engineering and wholly new approaches to the execution of the engineering process.

It is reasonable to anticipate that the new technology providing new capability will lead to a new practice of engineering. By a new practice is meant significant changes in how engineering is accomplished. The relative roles of men, machines, and organizations may be considerably different from that which we observe in engineering practice of today. Who does what, when, where, and how will change. Engineering will still be an art in 1976, but it will be quite a different art.

THE NEW PRACTICE—A NEW MAN

If we are to have a new practice of engineering in 1976 based on new capabilities, it follows that we may need a new kind of man to exploit these new capabilities and practice the new art. By a new man is meant one who is educated and trained for the new art and who has the talents and competence to use the new technology.

Engineering schools look ahead with respect to engineering science-oriented teaching and building on the science-oriented research. However, there is a tendency to look backward with respect to engineering practice-oriented teaching and to do little or no practice-oriented research. The art of engineering tends to be taught on a "this is the way it is done, boys" description of current and past practice. If the practice of engineering is going to change in the next decade, we have a responsibility to look ahead and prepare these young men for the future, rather than for the soon-to-be-obsolete methods of 1966.

KEY NOTES—KEY POINTS

A keynote address should presumably make a few key notes, or key points, to stimulate discussion during a conference. It is not always clear just what key points a keynote speaker is trying to make and, in order to bring mine into sharp focus, I will state them in summary form. My points are a combination of speculation, opinion, and proposal. I trust that you will not entirely agree with all those I am going to make because, if you do so, the keynote address will not have served its basic purpose—to stimulate discussion. To avoid the remote possibility that you might agree, I will deliberately state a somewhat extreme position which is sure to be somewhat controversial. You should bear in mind that I am speculating on engineering design practice as it might be in 1976 and not as it is in 1966.

The ten key points I would like to make are as follows:

1. Engineering design will be accomplished by an élite of gifted, brilliant, creative, highly skilled, individual professional designers—hereafter called "master designers."
2. The master designers will be concerned with generating design concepts and with deciding between significant alternatives for broadly defined design problems.
3. The master designers will work directly and interactively with highly responsive design systems based on powerful and large-scale information systems, one component of which will be ultra high-speed computers, but the heart of which will be information files and totally new kinds of software for working with these files.
4. The engineering design process will be formalized and organized into a highly sophisticated strategy for the optimum use of man and machine. The total design system will involve a highly sophisticated machine, a highly sophisticated man, and a highly sophisticated theory of design and decision-making.
5. All aspects of engineering design which can be programmed will be programmed, including aspects of design which at this point in time are not well structured or well understood, resulting in a significantly different design process from what we know today.
6. Communications, in the broadest sense of the word, will be at least as important as processing in the design system of 1976, and both will relegate calculation to a minor subset of total design activity. The conventional engineering drawing per se will cease to be the prime communications medium for engineering design decisions. Dynamic communications will replace static communications.
7. Design activity in an organization will be integrated with other decision-making and action-initiating activities, and will be conducted interactively with the total environment and operations of the organization.
8. The design systems for the late 70's have not yet been fully designed, nor have the designers to use them been designed. The initiative and leadership for accomplishing such designs should be taken by the engineering schools.
9. Engineering design education during the next five years should be highly research-oriented. The best design education we can give our students during the immediate years ahead is to make them junior partners with the faculty in conducting design research and development.
10. Future formal education for engineering design practice will require a study program through the Ph. D., plus additional requirements. The program should be highly selective and only those young men who are truly gifted for engineering design should survive. The design-oriented Ph. D. program for the future professional designer should emerge as the highest attainment in engineering education, involving all of the conventional requirements for demonstrated research capability, plus demonstrated original design capability.

THE PROTOTYPES

While the ten key points represent an extreme view, they stem from extrapolation to 1976 of experimental systems already under development in 1966. In particular, they are heavily influenced by current work on the development of the Integrated Civil Engineering System (ICES) underway at the MIT Civil Engineering Systems Laboratory and the large-scale, time-sharing hardware/software developments underway in industry and the universities, such as in MIT Project MAC. The 1976 versions of these prototype systems, even if projected in a very conservative way, provide the basis for the ten key points.

The prototypes for the hardware/software systems are under development. The evidence that the technology will be available is quite clear. The evidence that a man will be available to use this technology is not as clear. That the engineering organizations will be available is even less clear. There is very little evidence that organizations are responding to the implications of such change by advance planning and preparation. However, it is anticipated that the prototype organizations will emerge in the next five years, and competition will take care of the balance in the following five years.

RATE OF CHANGE—RESISTANCE

Change of any kind is always resisted by a subset of the people and organizations which are affected. The subset includes those who are are already content but who fear that their relative position will be adversely affected. Engineering is no exception. Many will adopt the attitude that the new capabilities are not needed. Historically, this has always been the case. (It wasn't too long ago that the log-table people argued that desk calculators were not needed, and it was just a few years ago that many questioned the need for computers in engineering calculations—some still do!)

There is no question but that the impending change in engineering practice will be resisted in a variety of subtle and not so subtle ways. There is also no question but that there will be pressures for change. The only question is how the resistance will affect the rate of change. The rate of change is a major unknown at this time. It is quite possible that we may indeed fall way short of the situation described in the ten points by 1976 if the anticipated resistance to change is sufficiently strong.

THE CHALLENGE

Our engineering schools can have considerable influence on the rate of change in engineering practice during the next decade. We can be an effective source of resistance and slow down the rate of change. Or we can accept the challenge of being an effective source of leadership. The choice is ours. This Conference on the Impact of Computers on Education in Engineering Design is particularly timely as there will be an impact in the immediate years ahead. We have the opportunity now to accept the challenge of being the designers of that impact and the instigators of change. The alternative is to be followers and to face a future of responding to an impact which results from external change. If we do not accept the challenge by conscious decision, we decide on the alternative by default.

Perhaps the most intriguing aspect of the challenge we face is that there are a variety of ways to pursue the challenge. Like all real design problems, there is no one solution. There are many solutions. Every school and every individual can participate in the search for good designs. We are sure to hear and see evidence of this during the conference. All of us have a role to play in designing the new practice of engineering for the 28-year-old engineers of 1976.

PAYNTER: *Taking Miller's fourth and tenth points I get the distinct impression that the person we are really talking about bears a strong resemblance to the person we have heard about in the past who was called the creative mathematician. I believe that creativity in basic science and in mathematics is not as different even now as you seem to make it out to be, or hint that it may be, from creativity in engineering design. And I definitely feel that, in the period ten and twenty years from now, the man whom you will screen out at this level of the Ph.D. and creative engineering design à la computer will be a man who simply has a tremendous, rational, creative gift. If you are talking about the strategy and the development of strategy, certainly no discipline has done more incisive, introspective looking at the strategy for creation of formal systems than mathematics itself. I would like to hear a response to this.*

MILLER: *I don't know whether I can give you a very eloquent response. I certainly will be willing to admit rather candidly that, as you know, we are trying to practice a little of what I've just been preaching in our own effort of taking a responsible role and attempting to design and develop these new design systems. Some of the most able and valuable young men working on that project actually did their undergraduate work in mathematics. The young men who are making the most notable creative contribution to the design of the ICES system are people who have a great flair for mathematics, and very abstract mathematics as far as that goes. I'm unsure of how far what you say is correct in terms of decision making. Even in 1976, with a very powerful design system of the type we are discussing, we are going to be working with incomplete and inaccurate information with problems for which there are resource constraints, that is, limits on the available amounts of money and time. In this whole area of compromise and trade-offs, I'm still betting rather heavily on the man to play a major role in making many of the judgment decisions, but he will be assisted in these by some very* *sophisticated, very formal strategies behind the scene represented in terms of the programming therein, which will facilitate his ability to make what may in the final analysis be judgment decisions. What's going on inside will, in fact, be a representation of a degree of mathematical sophistication that one would associate with the best of creative mathematics. What goes on outside when he turns around and walks away from that console, I think, will still represent characteristics that we associate with the engineer today, at least our image of a good engineer, that is, his ability to make decisions in the face of uncertainty and constraints and to do very well in the major decisions he makes.*

LICKLIDER: *I would like to add a couple of perhaps divergent comments here. First, I largely agree with Paynter that creative mathematics and creative design, within the limits of what I know about it, have a lot in common. Second, I want to suggest that if there is any thought in your mind that designers will be rather a drug on the market because only a few will be able to handle all the jobs, I would disagree strenuously because it seems to me that the search for solutions in problem space, or the search for good and reliable synthesis, is an extremely difficult and low probability pursuit. As soon as we have the apparatus, there are going to be facilities for doing a lot of that kind of thing. Then, I think, it will become important to do very much more than we now do, to carry planning which now, even at the international level, is about 1 1/2-or 2-step affair. Well, to carry design very much farther, into working designs and into planning, will take all the intellectual resources there are. One final thought is that if we ever do start to saturate what one might call modal design, that is, the finding of straightforward solutions to design problems, which I don't think we have come near yet, then there will be a great premium on achieving the odd-ball, but happily very good, designs that you can't get to along the standard channels, and that will readily take a lot of intellectual effort.*

MILLER: *Let me follow up on Licklider's comment by noting that at no place did I say we needed just a few of these master designers. By analogy, in the last ten years, with electronic computers we didn't just chew up in a relatively few milliseconds all the calculations that we would normally have done over many days. We simply went ahead and did a lot more computing than otherwise would have been possible. I think this is also true of design. Certainly we will do an enormous amount more of design, of a different kind and a higher level than we did before. I do feel that the effective design technician, while he will still exist and play a role,—and there will be relatively large numbers of them,—will operate differently in terms of where he fits into the picture. But the basic point is that the whole new technology is meaningless, or at least it does not achieve its full potential, unless we have the equivalent of the man I was labelling master designer to really master the technology that will be available.*

RESWICK: *It seems to me all the trouble started back with the Tower of Babel when people became unable to talk to each other. Licklider talked about the problem of design by teams, that is, by bringing various kinds of people together to achieve solutions. He also has had experience with group dynamics. I am thinking of what happens when an architect and an engineer get together to work cooperatively, to do creative design or innovation. It seems to me that Licklider was saying that the computer offers a solution to the Tower of Babel, that everyone will have to talk computer language and therefore will have to talk in the same language, and therefore will be able to understand each other, and that out of this can come optimum creative solutions. And yet, what actually does happen when such a group of people meets to do a creative job; and how does the best kind of idea come out of this group? So many things happen which are way beyond the information or the knowledge content of the people. What is the information content of creativity in a group, and when can a computer system ever reach the capability of transferring all of the things that must happen between people in such a situation to reach something like an optimum decision?*

LICKLIDER: *I certainly have no adequate answer or solution, but I would suggest a couple of things. First, I wouldn't hold that all the people would have to speak the computer's language. I think the trend is for the computer to speak the languages of the various disciplines. The second point would be that much of the misunderstanding between people from different disciplines is misunderstanding based on the fact that much of their talk is merely naming, and names are static. With the aid of the computer, one could substitute for the mere passage back and forth of static messages, the demonstration of behaviors. If you see the behavior, even if it's just pictures of behavior on a scope face, it's better than passing the words back and forth.*

APPLICATIONS IN ENGINEERING DESIGN

SECTION B

CHAIRMAN: F. H. BRANIN, JR., *IBM Corporation,*
Poughkeepsie, N. Y.

THE EFFECT OF COMPUTERS ON ELECTRONIC DESIGN

G. L. BALDWIN
Bell Telephone Laboratories, Incorporated
Murray Hill, N. J.

1. INTRODUCTION

Over the past decade, the computer has been playing an ever-increasing part in nearly all of our activities at the Bell Telephone Laboratories. Since these activities span a wide range of technical areas —from basic physical and mathematical research to the design of pushbuttons for telephone sets— it would be impossible to make a complete accounting of all the ways in which we have found the computer to be a useful tool. We will take two examples of practical design problems in the communications industry and give a brief history of how computer-aided techniques have developed over the years in each of these. By so doing we will be able to give a flavor of the manner in which the computer has changed, and is continuing to change, the way we do our work. The two examples themselves serve to illustrate the variety of our problems. The first —that of a communication transformer design— illustrates some of the problems we face in selection and design of components, while the second—that of designing large systems—shows what we need to do in order to assemble hundreds of thousands of components into working systems.

2. COMMUNICATION TRANSFORMER DESIGN

Quite naturally communications equipment is full of all sorts of broadband transformers, each of which must be designed to meet the particular circuit requirements dictated by its environment. The history of the development of computer techniques in this area is particularly enlightening for it shows how the computer has been used at many levels, ranging from specific one-shot calculations to a highly complex set of programs which virtually automates the entire process from design concept to the generation of detailed manufacturing information.

The word design as we here apply it to transformers is really a misnomer in that it encompasses a good deal more than physical realization itself. Also involved in the overall problem are the processes of:

Search: Can an existing transformer handle this job?

Synthesis: From the given electrical circuit requirements, what are the parameters of the equivalent circuit of the transformer?

Design: What are the physical quantities required to realize these equivalent circuit parameters?

Characterization: What is the total performance of the transformer we have either selected or synthesized?

Specification: What is the detailed information required for manufacture of this transformer?

Before the availability of the computer, these basic functions were carried out manually by reference to catalogs, rough slide-rule calculations, laboratory tests and drafting. We will now attempt to show how, over the years, increased use of the computer has gradually automated each of these functions.

Analysis

The first use of a computer in the design process is typically that of analysis: How will a given device or system behave under various conditions of operation? The first step in performing such analysis is to develop a mathematical model which suitably represents the physical situation. This requires a combination of intensive theoretical and experimental studies. In some cases—as in linear lumped-parameter filters—the approximation is very close, and one is able to make calculations with confidence that they will be accurate within a small fraction of a percent. In others—as in modeling the behavior of a typical modern central office—we are unable to develop a model simple enough to handle on a computer which is, in turn, sufficiently realistic to provide more than fairly gross approximations to the system's real behavior. In our example of designing broadband transformers, the model is in the form of an equivalent circuit made up of linear lumped-parameter elements.

Next, one must develop a method of relating the values of the parameters of the mathematical model to the physical characteristics of the device or system being studied. Similar relationships between the behavior of the model and that of the physical system are needed in order to be able to interpret the results of the study. In our example the physical characteristics of the transformer are core size and permeability, wire size, turns ratio, insulation thickness and several others. The parameters of the mathematical model are the values of the elements in the equivalent circuit, and in this case the behavior of both the model and the device is either the transient response or the frequency characteristic curves, depending upon the transformer's application. Since our emphasis here is more on the design process itself rather than on broadband transformers, we will not go into detail as to the specific equivalent circuit which was used.[1]

In the first step of introducing the computer to the problem of design, we program it to accept physical parameters of a given transformer, calculate the values of the elements in the equivalent circuit, and then perform standard network calculations on that circuit. This allows an engineer to select a set of physical parameters, examine the resulting performance, and zero in on a satisfactory design by repeated runs, varying the physical parameters on a cut-and-try basis.

Synthesis

One would initially suppose that the problem of synthesis would simply be a matter of reversing the problem of analysis, that is, solving the equations for the physical parameters in terms of the desired characteristics. However, this is generally not thc case; there are only a small number of physical parameters which can be altered and one usually has a larger number of external characteristic values to be satisfied. Thus a good deal more investigation, both experimental and theoretical, is needed in order to determine what specific external measurements are of greatest significance, so that the inverse solution can be made possible at all. We are generally faced with a set of non-linear equations which defies solution even with a computer at hand. If we are fortunate, some further analysis will produce a few gross relationships which allow at least an approximate solution to these equations. Now we are faced with approximate answers to a mathematical model which in itself is an approximation of the actual device. The next step is utilization of the actual device. The next step in utilization of the computer is to write a program which accepts desired external characteristics as input and uses the rough approximations to estimate the physical parameters, providing input directly to the analysis program.

This then allows the engineer to zero in on his final design with fewer computer runs. It also shows us the start of a system of programs rather than of a single one-shot effort. The next three stages of development in our transformer design example are somewhat independent—one aids the designer in his catalog search (after all we may already have a transformer in stock which fills the bill), the second generates the detailed manufacturing information required by the factory, and the third allows his cut-and-try process to take place more conveniently and far more rapidly.

Search

Once a suitable model of our transformer exists, then a specific transformer can be exactly described by a set of numbers. If we have recorded all previously designed transformers in computer memory (typically on a magnetic tape "catalog"), we are able to match the parameters of the desired transformer against those for which designs already exist. Should any fall within suitable tolerance, then the analysis program is brought into play and the designer can see how well any existing design fits his requirements. If there are any that come close to fitting the requirements, they are so indicated and their complete characteristic curves are presented to allow the engineer to see if one of them fits the bill. Otherwise, we go through the synthesis phase (and, of course, add this new transformer and its characteristics to the catalog).

Output

Once a design has been completed, the engineer has (as a direct result of the process described above) a complete functional characterization of the transformer. But the computer now "knows" all of the physical quantities required for actual manufacture. Therefore, a part of our growing design system is the production of manufacturing information in a format which is directly acceptable to the factory.

Let us see what the computer has provided us with so far. Our program system provides the engineer with a significant saving in time, cost and accuracy. The system, through its search routine, will minimize the number of new designs. If a new design is necessary, the frequency band of interest is centered, thereby providing greater possibility for multiple-use transformers. This centering procedure will tend to minimize overlap in designs; hence, a tendency toward fewer different designs. By filling out the transformer design request, the engineer can have within a day's time complete information on any transformer which meets his needs and/or design information, if it is possible to produce such a transformer.

Although we do indeed have a system of programs, it was designed in such a fashion that part of it may be used independently. For example, both the synthesis-design and the analysis portions have found extensive use as individual programs.

Direct Interaction

About the time that our transformer select-and-design system became operational, a completely new

[1] For a treatment of the broadband transformer design problem in greater detail, see "Communication Transformer Engineering by Computer," by C. M. Bailey, F. J. Kasper and D. Katz, in the Proceedings of the 1965 Electronics Components Conference.

facility was made available to us at the Laboratories on an experimental basis. The present talk is not intended as a description of that facility, for other talks in this session will go more deeply into similar developments. Briefly stated, this equipment, known as GRAPHIC-I, provides a graphical input/output capability whereby a user can communicate with a large computer via a CRT display and a light pen. Physically it consists of a CRT, a keyboard, a light pen, and a smaller computer (a PDP5) with an added memory, plus hardware which allows data transmission to and from the 7094. It differs from other similar devices (such as SKETCHPAD and the General Motors DAC System) primarily in that the local computer and its added memory provide a good deal of stand-alone capability. Thus communication to and from the 7094 tends to take place relatively infrequently but with fairly large chunks of data handled in one interaction. GRAPHIC-I is really an experimental vehicle leading to the design of GRAPHIC-II, which will be used to provide graphical communication with our GE 645 system about a year from now. Its major limitation is one based on 7094 software—we didn't feel it worth while on our particular time scale to provide capability for interruption of the 7094 on other than a between-job basis. For this reason, one might have to wait up to a few minutes to capture the 7094, but from then until the transaction is completed, he commands the 7094's full attention.

So much for the equipment. How has this changed our outlook on design in general, and on broadband transformer design in particular? Let's review the basic method of design we have used all along the way. Essentially it has been to guess what the proper physical configuration would be, analyze this and then make another guess. The computer's role has been partly clerical and partly calculational. It has been clerical in that the program first looks in a catalog to see if a new design is really needed. It has also been clerical in that it "remembers" past guesses and their consequences and in that it presents the finally agreed-on design in the proper form for manufacture. The calculational roles of the computer have been to perform the analysis and to assist the engineer in making better-educated guesses. Now what the GRAPHIC facility provides is greater simplicity of use (the engineer no longer has to punch cards and assemble mysterious decks in the proper order)[2] and an order of magnitude in compression of the time scale. A complex design might require a dozen or so attempts—each depending heavily on the results of former trials. Using typical batch processing procedures, this might stretch the overall time between origination of the needs and realization of a satisfactory design to several days. Of course, during this time period the engineer is presumably off doing something else, but the turnaround time is still measured in days. Having a directly interacting facility for entering data, viewing results and entering new data can reduce this to a matter of minutes.

While we are on the subject of electronic components, we might profit by looking into another application of graphic consoles. A large part of what we generally refer to as design consists of looking through catalog information for a component which meets various combinations of requirements, including electrical and physical characteristics, costs, reliability, and numerous others. Although we have not yet built up a file, we have developed a structure for such an application and have played with it enough to be convinced that this will be of great value. What we are really doing is searching a huge file structured in the form of a tree, and at each node asking the user which branch he wishes to take. Although the identical searching process is feasible using a typewriter as the input/output device, the CRT light pen is far faster in displaying the information pertinent to each node, and it is far easier for the user to point to something by a light pen than to type his request out.

3. SYSTEM DESIGN

In addition to designing components we are, of course, interested in gathering these components into various levels of sub-assemblies which are built up to form large systems. It is not at all unusual today to discuss a system with nearly a million electrical components and hundreds of thousands of interconnecting wires. In this area, I am sorry to say, we can't point to as neat and orderly a growth of automation as in my first example, although there have been a few notable approaches to complete design automation systems.

When we look at system design, the first problem that faces us is the sheer bulk of the data, together with the complexity of the interrelationships among subsets of the entire data. For this reason, most design automation efforts aimed at large systems have first of all attacked the clerical, record-keeping problems rather than the activities involving some creativity. Automation of record-keeping is virtually mandatory here, not only from the point of view of economy, but also in order to provide the host of design engineers up-to-date and consistent information as to the current status of the system. Furthermore, one can ask questions about the ap-

[2] At present he has to learn other mysterious things like pushing "light buttons," waving the light pens and the like. Our ultimate aim is to make the system as simple as possible for the user.

proximate cost of the entire system or of a subsystem, or find out how many times and where a particular component is used, and receive answers economically and quickly. With manual methods of keeping records, either of these questions would require so much thumbing through stock lists that the answers would cost too much, and by the time they were generated they might well be out of date.

It is appropriate here to point out one of the major problems we have encountered in using the computer to document large systems. Especially in engineering development efforts, the nature of the information we wish to keep is continually changing. Furthermore, the various people involved with the job are interested in different subsets of the entire information. However, the data-processing software we have available to us today renders changing the format of a file very difficult, since it necessitates changing each individual program which works with the file. Thus we are faced with a combination of two problems: a good deal of our programming effort is wasted by trivial but time-consuming editing to introduce changes in file or input/output formats; and many potentially useful ideas are abandoned because of the cost involved in implementing them.

A second area in which automated design of large systems has suffered stems from a heavy dependence of most of them on the detailed nature of the equipment being designed. That is to say, things like the type of logic used and the size and terminal identification of sockets tend to become so deeply ingrained in these programs that we almost have to write a complete new system of programs to handle a new line of equipment.

So much for the negative side. There are a dozen or so companies now designing and manufacturing large electronic systems, and each has its own computer-based record-keeping system. Each of these provides to one degree or another a voice of conscience for the designer, checking his design against various rules of fan-in, fan-out, impedance matching and the like. This, incidentally, has proved to be one of the most valuable of the services provided by the record-keeping type of program. I feel with a fair degree of confidence that the problems relating to software and standards will soon be solved to the point where we will have really flexible engineering data-processing systems.

What now about the design of large systems? There are some design problems which are common to nearly every large system, and some are unique to certain specific technologies. In this review we will look at a few of the common ones.

Perhaps the most worked-over problem in large system design is that of placement. This might mean location of components on a printed circuit board, location of boards on a frame, or even location of equipment frames in an office. Many attempts have been made to do placement entirely automatically. One technique developed at Sandia involves establishing artificial force fields among the components.[3] Another developed at the Bell Laboratories involves a rough placement of clumps of components, followed by an iterative scheme of interchanging pairs of components and determining the resulting effect. I know of other techniques and I am sure there are many more of which I have no knowledge. Of one thing I am sure, however: that is, that none have been found to be entirely satisfactory in a practical system design situation.

Let us observe that the previous paragraph discussed a problem in automated design without mentioning any criteria by which a given design can be evaluated. This is really the heart of the matter. The most easily stated criterion is minimum total wire length, and even in this case no completely automatic techniques have been found to solve practical problems. But the problem is far more serious in that most design engineers are unable to specify all of their criteria at all clearly. In some cases crosstalk is important, so that there exists a limit on the distance over which a pair of wires can run parallel to each other. In other cases, capacitive loading or delay are critical, and here one wishes to minimize the length of the longest lead in a certain class. Finally, esthetic considerations come into play, primarily as an aid to the maintenance man. If we have a structure (say a shift register) which has a definite electrical pattern repeated many times, it is desirable to have the corresponding physical pattern repeated similarly.

For two reasons, then, most of us have given up on fully automatic placement, and have turned to use of graphic consoles operated by the design engineer himself. Here the computer is able to keep accurate records and remember, and cause to be repeated, specific patterns; it can also evaluate a given layout with respect to any of a number of different criteria. We leave to the judgment of the man, then, those criteria he cannot really explain explicitly.

Another closely related problem is one we refer to as treeing and routing of wires. Given the electrical schematic and the position of each component, we know which terminals must be connected together. In this instance, completely automated methods have been developed for determining the proper order of connecting the terminals to give minimum wire length, and also for finding a route

[3] "ACEL" Automated Circuit Card Etching Layout," C. J. Fisk, D. D. Isett, Proceedings of the 1964 SHARE Design Automation Workshop.

for each wire such that no rules related to bunching of wires are violated.

Now let's look at the early stages of designing a large system. To my knowledge, each effort here has been of a one-shot nature. It is not unusual to see two kinds of simulation used in the early stages of development of a project. The first of these is a fairly gross overall representation of the system to allow the designers to find out where the major flows of information will occur and which are likely to be the most critical parts of the system in determining its ultimate performance. The second type of simulation is to take a small chunk of the system and simulate its behavior on a pulse-by-pulse basis. As far as I know, even our fastest computers are too slow to allow a complete, detailed simulation of a sensibly large system. Indeed, even if this were possible, it is hard to imagine what one would do with all of the numbers which would result.

In the area of detailed logic design we again look to the graphic console to be of great use. The day is not far off when a designer can sit in front of a console, draw chunks of circuitry, asking the computer to repeat given chunks when a repetitive circuit comes up. The machine will then check the circuit against the design and manufacturing rules appropriate to the particular technology and tuck the information away for recall at any later time either by the designer, by one of his colleagues, or by programs used later on to generate the added information required to manufacture the system.

4. CONCLUSION

Computers have already become indispensable in the design of communications components and systems. Today the uses are largely of a clerical nature—receiving information, massaging it gently, and printing it out in various forms. However, the problems are rapidly being solved to the point where we are looking to the design process itself. In some cases we are able to develop algorithms which solve a design problem completely. In most cases, on the other hand, we look to the development of programs involving direct communication between the designer and the computer through on-line graphic consoles.

ACKNOWLEDGMENTS

Practically none of the work described here can I claim as my own. In particular, the use of graphic consoles in general has been carried out by several people at the Laboratories who are too numerous to mention. The use of these consoles for transformer design and file searching has been done primarily by Messrs. P. G. Dowd and T. J. O'Conner.

HART: *I would like to have your opinion about knowing something about computing, but not having any courses in it. Would you explain it a little?*

BALDWIN: *I think learning how to use a computer is becoming easier and easier. I think that if a student were to spend a lot of time learning Fortran, Algol, Flugol, or what have you, and also the mechanism of coding, he would be wasting his time. There is a need for the high priests of computing who will supply the eventual software whereby the computer will talk people's languages; but a man who is going to be a designer needs to know very little about computers. What he needs to know is a lot more basic mathematics and physics.*

CONWAY: *You, of course, are aware that almost all engineering students are now learning to solve problems with a computer and understand how to use it. My observation is that if they don't learn it in school, once they get into industry it is very difficult for them to learn it.*

BALDWIN: *I would like to address myself to two points. First, today you need to teach some programming because we have not reached the utopian situation where it is easy to use the computer. But, to me, the reason you teach programming is so that students can use the computers in their other courses to develop a good fundamental education.*

Now, my particular job is hiring people who will become these high priests and supposedly make the computer easier to use, but we're making it more and more difficult. The last machine I really understood was the IBM 650. But when I hire people, I do not care whether or not they have had any programming experience in school, and I run a programming department. What I want is good, fundamental mathematics and physics and a statement of interest in this particular field, because people can pick up programming easily if they are going to be good at it. They can pick it up certainly in a year on the job, and we're hiring these people for 40 years. A second point is that schools have machines which are far behind the ones that are used in industry. So you get a fellow that says "Ah ha, I know how to program the Blubiac," and this doesn't do him any good at all because we've got something else.

BRANIN: *I'd like to make a comment on that point. I feel that once an individual has learned to program in any machine language, or any language at all, he has mastered the fundamental principles. The translation to another language is nowhere near as difficult as learning the first process.*

BALDWIN: *Maybe he learns to mistrust the computer. This is indeed a lesson that one must learn.*

LICKLIDER: *I'd like to make an intermediate proposal because I agree with you that there shouldn't be courses in computers which spend three weeks on binary notation and teach how to program in some particular machine language. I also agree that people can learn pretty quickly on the job, but I notice that the self-taught people miss something very important. They haven't really learned to understand about formalisms or the syntax of computer languages, and they don't have a set of the abstractions without which they cannot really make any progress in the modern world of computers. So I would argue that there is a course to be taught about computers; maybe it isn't being taught anywhere yet, but there is some lore somewhere between computers and mathematics that would be awfully good to have.*

HART: *It is very easy for a student to go through four years of engineering school and never solve a problem. I think one of the opportunities the computer provides is that the student, by getting involved in the sweat and tears of writing a program to solve a problem, develops the concept of the problem-solving procedure, the idea of an algorithm, and I think exposure to this early in his educational career is extremely important. I don't suppose that you must then continue with more how-to-do-it courses, but give enough training in programming so that a student can approach a computer and solve a problem with some insight into what this involves.*

BALDWIN: *One of my reactions is that he can learn the problem-solving procedure, which is actually mostly problem formulation and not problem solving. Typically, you're reformulating the problem as you solve it, whether you use a computer or not. But can't he do this by standard analytical methods? You don't think so.*

BRANIN: *There is a strong point that Hart brings out, which is that the discipline of programming a problem forces you to learn it in more detail and more thoroughly than you would otherwise, so I would agree with him on that.*

SHANNON: *With the introduction of our time-sharing system, we have moved in exactly that direction. We moved the teaching of computer programming into the freshman mathematics course, deleted a course in engineering in "Introduction to Computation in Numerical Methods," and inserted a new course in "Field and Electricity and Magnetism." This is the current direction in which we're moving.*

BALDWIN: *Well, that sounds good. I don't believe everything I say myself. I think certainly this is a very serious problem in education.*

STRUCTURAL ANALYSIS BY FINITE ELEMENT METHODS

M. C. C. BAMPTON
The Boeing Company
Seattle, Wash.

I. INTRODUCTION

The intention of this paper is to discuss only one aspect of aeronautical design procedure which is influenced by the existence of digital computers. This particular aspect is the analysis of structures. While limiting the technological scope of the discussion, this topic provides an excellent illustration of the dramatic development of an analysis capability to cope with a very wide range of problems. Such progress would have been impossible without the associated development of digital computation machines.

During the last decade the aeronautical industry has been mainly responsible for the introduction and establishment of finite methods. These methods have been denoted by various names depending upon whether they have inherently regarded forces or deformations as the unknowns of an analysis, or whether the method has been expressed solely in terms of matrix algebra or has implied certain computing algorithms. These methods all possess one feature in common. That is, they depend upon the process of idealizing the structure into an assembly of finite elements which are analytically connected at nodes or boundaries.

Examination of the literature will show that development of the finite element method is continuing vigorously both within industry and at universities. Effort is being directed along two main avenues of development: improvement of the element and improvement of the system that accepts the element. The former implies more precise representation of technological behavior and the latter implies more efficient computing procedures.

The aircraft structural problem which can be said to deserve the strongest claim to having prompted the development of finite element methods is the multi-spar, multi-rib swept wing. This structure possesses both complexity and a high degree of indeterminacy. A decade ago, such a problem was a challenge to analysis; today, finite element methods have made its solution commonplace. It is also evident from reports that much experience has been gained in applying such methods to other aircraft structural analysis problems. These applications of finite element methods reflect advantages over classical aircraft analysis procedures in that they reduce considerably the weight penalties associated with making assumptions in idealization and that they are much more capable of producing refined descriptions of complex structural behavior. This paper will describe several typical applications and, where possible, show the correlation between test results and analysis.

It is believed that there will continue to be progress made in developing and applying finite element methods of structural analysis. It is also believed that considerable progress will be observed in what might be regarded as a lateral direction, that is, the application of such methods to other technological areas. This trend can already be evidenced with the molecular vibration analysis by Kaufman and Kaufman,[12] the seepage analysis by Zienkiewicz[9] and temperature analysis by Visser[13] and Bampton.[14] To lend weight to this contention, two applications of relevance to aircraft design are included in this paper.

Finally it can be said that, just as the finite element methods are dependent upon the digital computer machines for their existence, so are they also dependent upon the appropriate education of engineers for their correct application, interpretation and development. The overwhelming response to related courses being offered within the industry is a clear indication of the need for such education.

II. FINITE ELEMENT METHODS

It is not the intention of this paper to discuss finite element methods with any thoroughness, since this has been successfully accomplished elsewhere in the literature.[1-8, 10, 15] The objective is to dwell upon applications and thus stimulate interest in the methods. However, some statements concerning concepts should prove beneficial within this context.

Before proceeding to a finite element analysis, an engineer must have at his command within the computer capability an array of mathematically defined structural elements. The force-deflection relationships of each element must be explicitly expressed. An array of typical linear isothermal elements is shown in Figure 1. The existence of such elements will enable an engineer to idealize a structure into an assembly of such elements. An idealization is shown conceptually in Figure 2 in which the plan view of an aircraft swept wing is covered by an analysis grid. The grid, in turn, implies the location of analysis nodes and finite elements.

The forces and deflections of the elements are

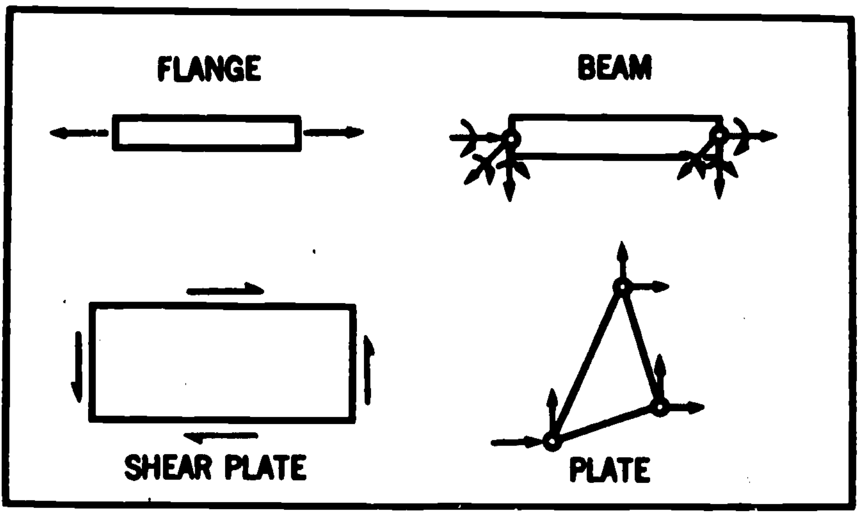


FIG. 1 TYPICAL LINEAR ISOTHERMAL ELEMENTS

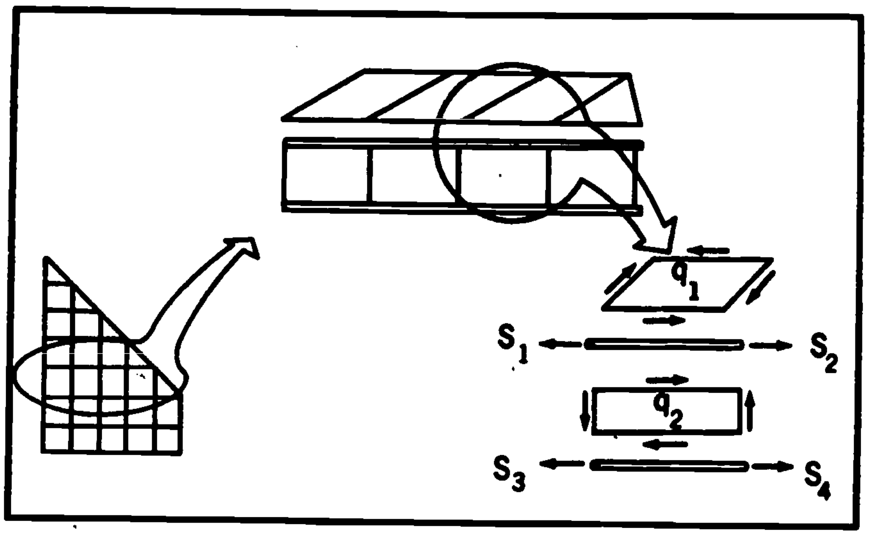


FIG. 2 IDEALIZATION INTO FINITE ELEMENTS

then expressed in a set of linear algebraic equations such that equilibrium and compatibility are satisfied at the nodes and boundaries of the structure. For a large complex structure, this leads to a large number of simultaneous equations. In realistic problems, many hundreds of equations can be involved. Thus it becomes clear that for an engineer to merely discuss his problem he must resort to the compact notation of matrix algebra, but that to solve it, he must rely upon the availability of a digital computer.

III. APPLICATIONS OF THE FINITE ELEMENT METHODS

1. Airplane Design Studies

Extensive application of finite element methods occurred during the design of the Boeing Model 737 Transport Airplane. A representative selection of the studies is shown in Figure 3 and their pertinent features will be described. In all cases, the computations were automatic and the engineering effort arose in connection with the interpretation of results and preparation of data for idealization. It is in the two latter phases that engineering judgment must be exercised if an analysis is to be successful.

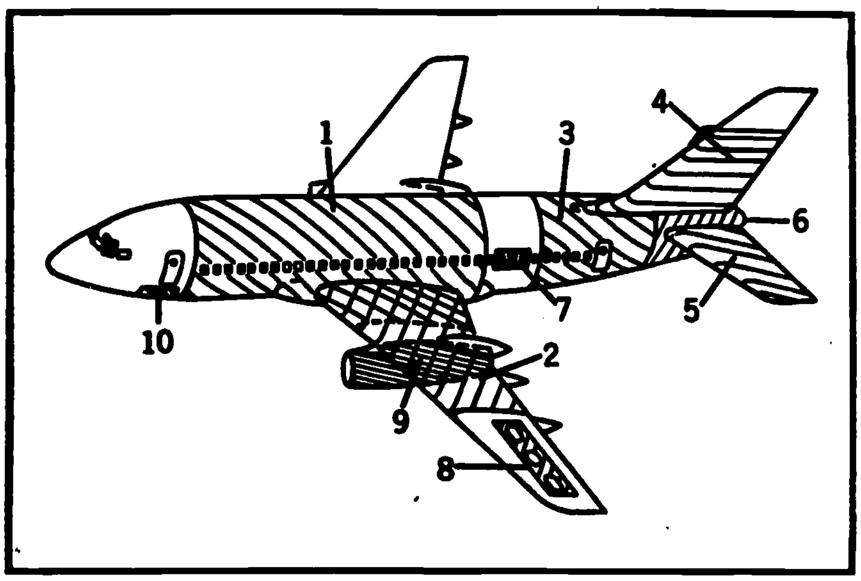


FIG. 3 BOEING MODEL 737 DESIGN STUDIES

Referring to Figure 3, Region 1 represents an analysis of the complex wing-fuselage intersection structure. This area, with its conglomeration of bulkheads, flexible frames, cut-outs, fuselage cross-section transitions and wing-fuselage interaction structure, contains a formidable challenge to classical analysis procedures even when supported with empyrical data. Region 6 has similar features to those possessed by 1. Region 2, 4 and 5 are wing structure and inherently contain an analysis problem due to their being swept. For a two-spar structure this is the shear transfer of load from the front to the rear spar and the amount can not be determined by classical methods unless supported by empyrical data. Figure 4, a region on the Boeing Model 727 main wing, shows the excellent agreement between stresses obtained by finite element analysis and stresses obtained during static tests on the airplane. Regions 3, 7 and 8 of Figure 3 represent cut-out analyses around doorways, windows and access panels respectively.

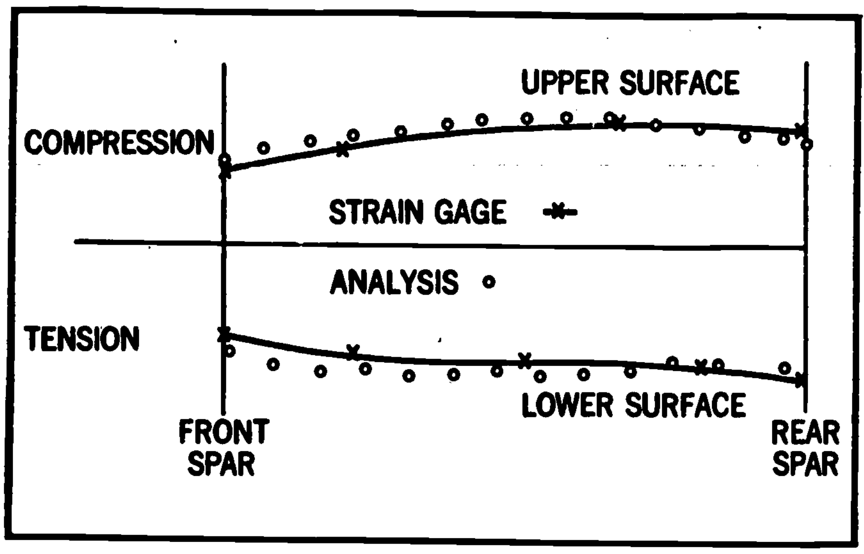


FIG. 4 WING ROOT SPANWISE STRESSES

One major consideration in such analyses are the boundary conditions for the regions of analysis. These are usually obtained from a prior gross analysis or by invoking Saint Venants Principle. Region 9 involves ducting around the engines and requires consideration of thermal effects. Region 10 represents an analysis that is small by comparison with those previously mentioned and yet contains enough indeterminate elements to justify the use of a finite element method. The problem concerns interactions of floor beams and door sills.

In order to give some meaning to what might currently be considered a *large* analysis, one of the preliminary design studies performed for the Boeing version of the C5A Airplane is shown in Figure 5.

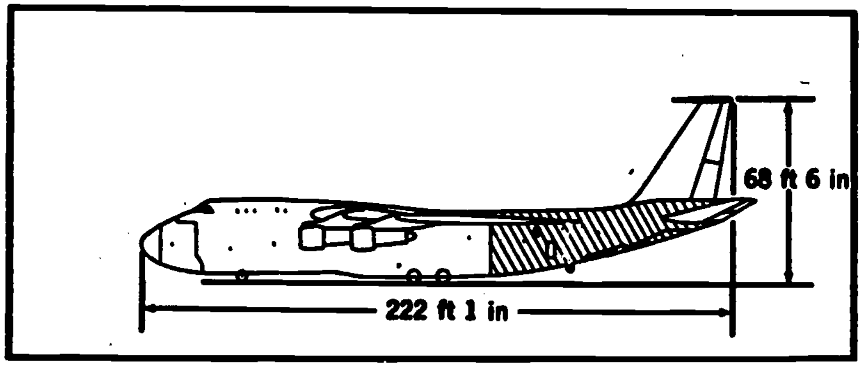


FIG. 5 C5A PRELIMINARY DESIGN ANALYSIS

This region of fuselage includes large access openings for loading vehicles, etc. The analysis grid is indicated in Figure 6 and the details that describe the size of the problem are included in the figure.

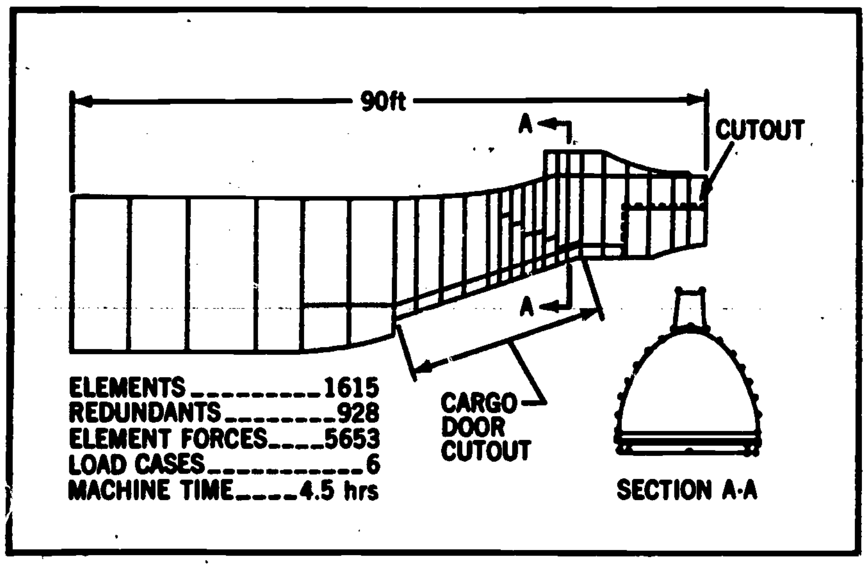


FIG. 6 C5A REAR FUSELAGE ANALYSIS GRID

2. Non-Linear Material Behavior

In assessing the fatigue and fail-safe characteristics of an airplane component, a preliminary phase involves the determination of stresses that are distributed around an existent crack. A typical component to receive such attention is a panel comprised of cover plate and stiffeners. It is usually assumed that in the middle of the panel there exists a crack of finite length and zero width. In other words, the panel possesses a structural discontinuity additional to those caused by its attachment to stiffeners. To solve the stress distribution, finite element methods are used.

It is known that for significant loadings on the panel, the concentrations will cause the stress within a certain region to exceed the elastic limit. Thus nonlinear material behaviors have to be represented. These can easily be read into the computer in the form of a table that relates stresses to strains for particular values. A typical curve is shown in Figure 7. The solution, which follows the approach

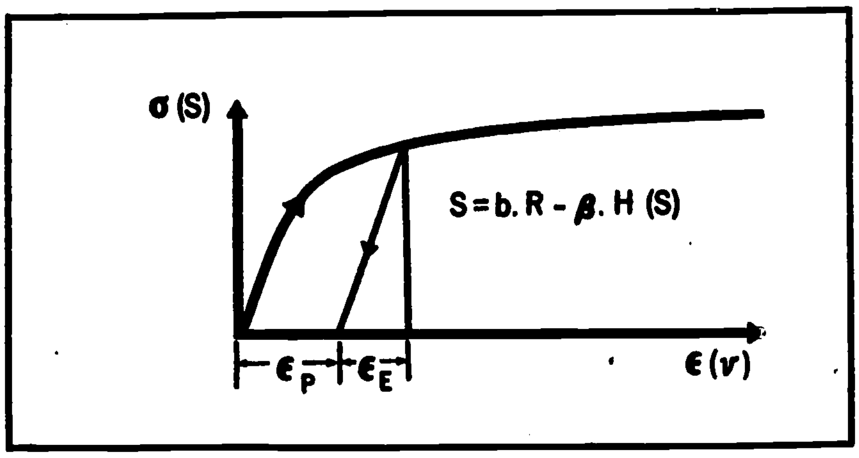


FIG. 7 NON-LINEAR STRESS STRAIN REPRESENTATION

outlined by Argyris,[14] uses an initial strain concept, and involves both iteration and inversion processes.

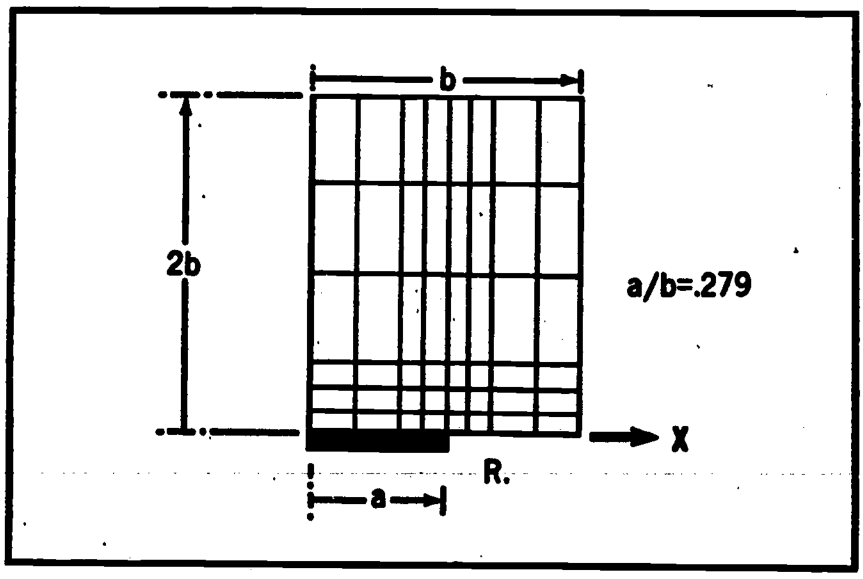


FIG. 8 CRACKED PANEL ANALYSIS GRID

An example of a simple problem is shown in analysis grid form in Figure 8. The plate is uniform, loaded by a uniform stress along the edges parallel to the crack and, since it is doubly symmetric, only one quarter need be analysed. The distribution of grid lines reflects an anticipation of the location of the stress concentrations. The grid lines define the positions of flange elements and the rectangular spaces imply uniform shear fields.

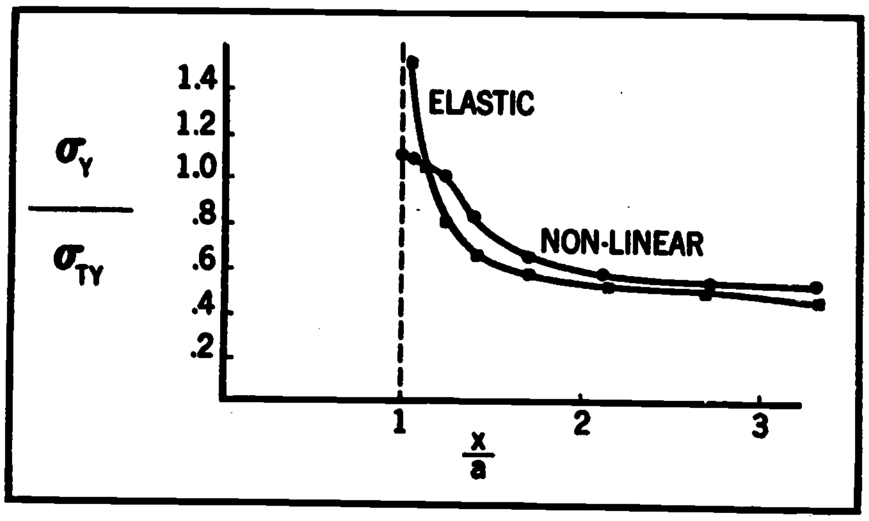


FIG. 9 CRACKED PANEL RESULTS

In Figure 9 can be seen the results due to a certain magnitude of applied stress. The stresses shown are those acting normal to the crack at the line co-axial with the crack (Y=O). Assuming first a fully elastic material, there can be a very high magnitude of stress near the crack tip. This corresponds to the singular point that would occur in an analytical solution. Allowing the material to have non-linear properties shows a reduction in peak magnitude, yielding and general redistribution.

3. Engine Cowling Analysis

The engine cowling structure was analysed by a finite element method and certain critical deflections obtained. The cowling was a tube constructed from titanium honeycomb panels and the tube was loaded by internal pressure. Analysis was made complex by virtue of the irregular shape of the tube cross-section, the presence of plates (splitters) connecting opposite sides of the tube, and the need to represent the bending behavior of the honeycomb panels. A view of the cowling is shown in Figure 10 and part of the analysis grid and the pressure loading is shown in Figure 11.

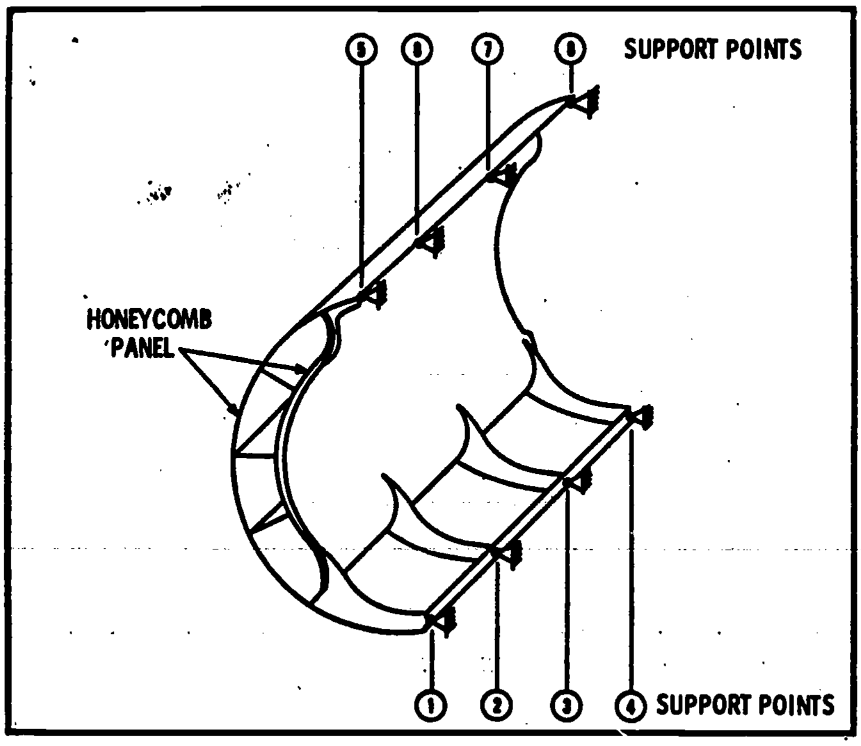


FIG. 10 ENGINE COWLING

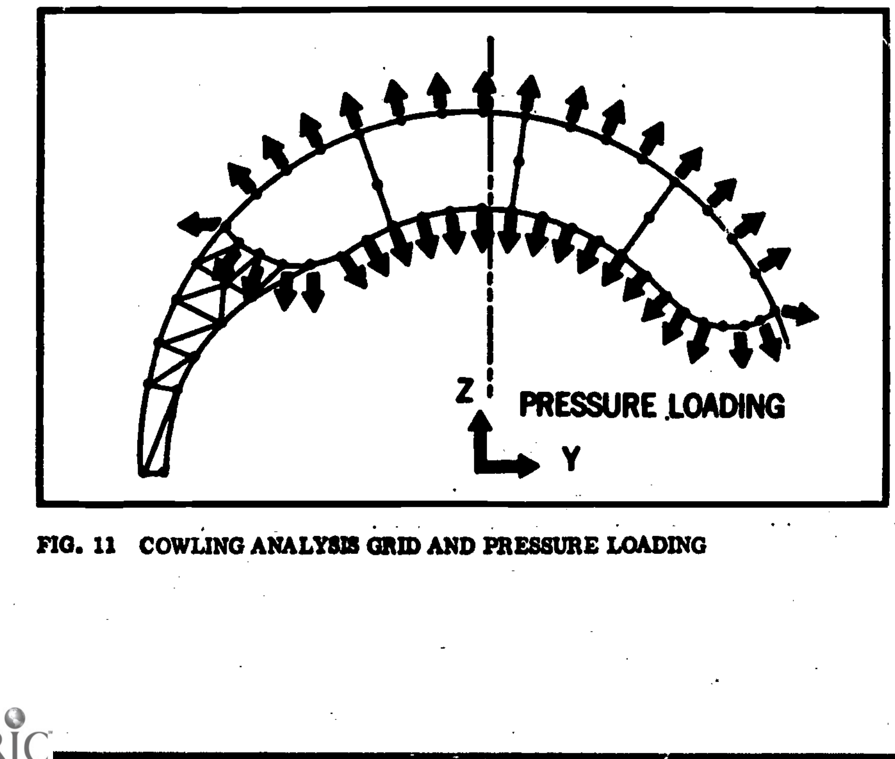


FIG. 11 COWLING ANALYSIS GRID AND PRESSURE LOADING

The finite element method used to analyse the cowling structure included a triangular plate element that possessed bending and membrane properties. This was used to represent the honeycomb panels in the idealization. The computing details of this analysis, which produced both stress and deflection distributions and an influence matrix, are as follows:

Number of nodes	=364
Number of structural elements	=401
Computing time (IBM 7094 Mk II)	= 45 minutes

Tests were performed upon the cowling and deflection measurements obtained. A set of these results is shown in Figure 12 for one cross-section of the structure. It can be seen that the finite element analysis predicted the test results with considerable accuracy.

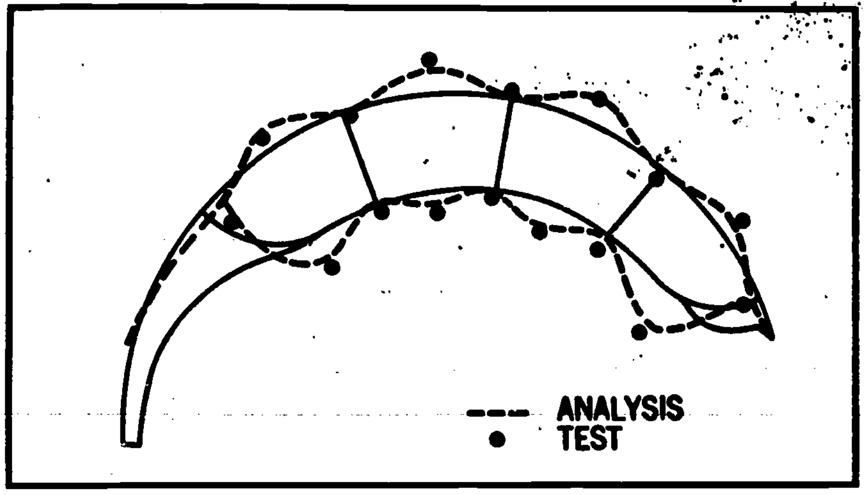


FIG. 12 COWLING DEFLECTION RESULTS

IV. THERMAL AND ACOUSTICAL APPLICATIONS

The concepts expressed previously with regard to structural analysis by finite element methods are valid also for analysis in other technologies. In the matter of analysing temperature distributions, it can be shown that for linear conditions a direct analogy exists between the formulation of the structural and thermal equations. Thus one solution program could solve both classes of problems, provided that the appropriate data was supplied. In the acoustical problem, a fluid element is treated as though it were a structural element and is therefore easily incorporated into the structural solution.

1. Thermal Analysis

Using a variational approach, Visser[13] has shown that a close analogy exists between the solution of conduction problems in thermal analysis and the solution of structural problems. Thus it is possible to conceive of finite elements representing assumed temperature behavior and expressed mathematically by the corresponding heat flow-temperature relationships. In matrix form this is denoted: Where $\delta = \emptyset . T$

δ=Column matrix of nodal heat flows
T=Column matrix of nodal temperatures
$\emptyset$=Square matrix of influence coefficients for a thermal element

In the analogous structural equations, the matrix $\emptyset$ would correspond to the stiffness matrix of a structural element.

Visser explained the $\emptyset$ matrix for a flange and the $\emptyset$ matrix for a triangular plate is derived in Appendix A. Using similar approaches, mathematical expressions for a comprehensive array of idealized thermal elements can be constructed. This array could match the types of elements used in a structural analysis.

If it is possible to idealize a thermal problem into a set of mathematically defined finite elements, it remains for a solution procedure to assemble and solve for the unknowns. The method used can be identical to that used in solving a structural problem, but will not be discussed in detail here.

The applications of thermal conduction analysis using the triangular plate element will be described. The first is a demonstration of Saint Venants Principle. Irregular distributions of temperature were imposed on the boundary of a plate that was "long" in one direction compared to the other direction. The grid used is shown in Figure 13 and the boundary condition of the three remaining edges is such that there is zero heat flow across them. Interior nodes also have zero resultant heat flow.

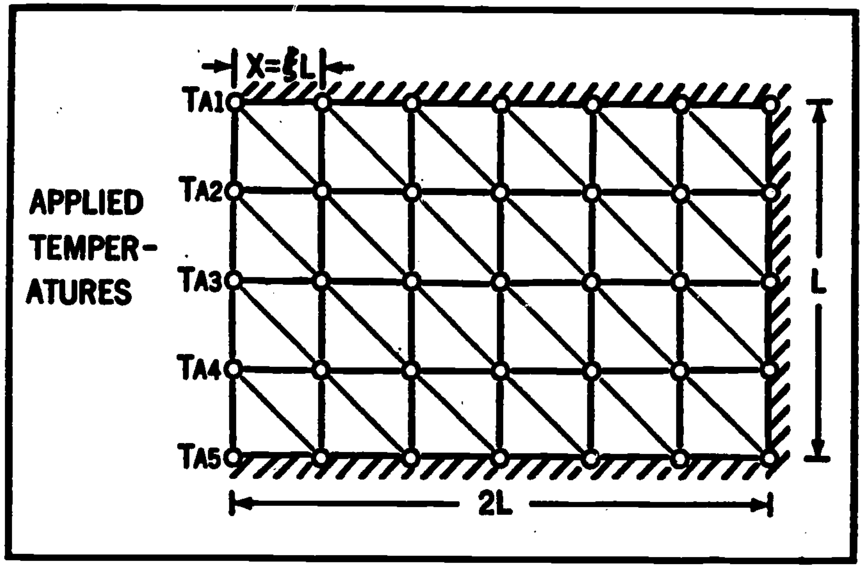


FIG. 13 SAINT VENANT THERMAL ANALYSIS GRID

The results of imposing two sets of temperature conditions are shown in Figure 14. Plots of temperature distributions across the width of the plate are given at various distances (ϑ) from the application edge. It can be seen that the irregularities have smoothed out to give uniform average values within two width-distances of application.

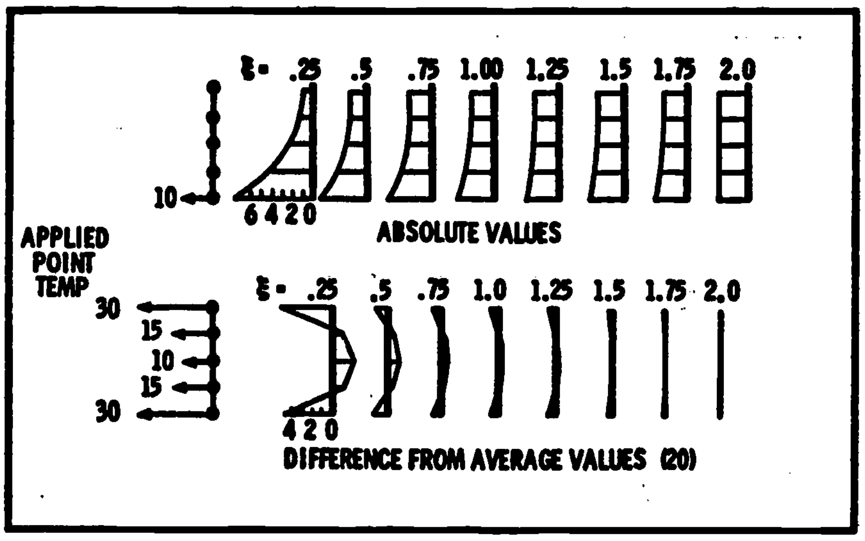

ξ= .25 .5 .75 1.00 1.25 1.5 1.75 2.0
10
6 4 2 0
ABSOLUTE VALUES
APPLIED
POINT
TEMP
30
15
10
15
30
ξ= .25 .5 .75 1.0 1.25 1.5 1.75 2.0
4 2 0
DIFFERENCE FROM AVERAGE VALUES (20)

FIG. 14 SAINT VENANTS PRINCIPLE - TEMPERATURE DISTRIBUTIONS

The second application is shown in Figure 15 and was performed to test correlation with another method.[21] The exact analysis involved a solution to the heat-conduction partial-differential equation and the matrix analysis used finite elements. A linear temperature distribution was imposed along one edge, zero temperatures maintained along two others and zero heat flow along the remaining edge. Plots of temperature distribution show that correlation between the two methods was excellent.

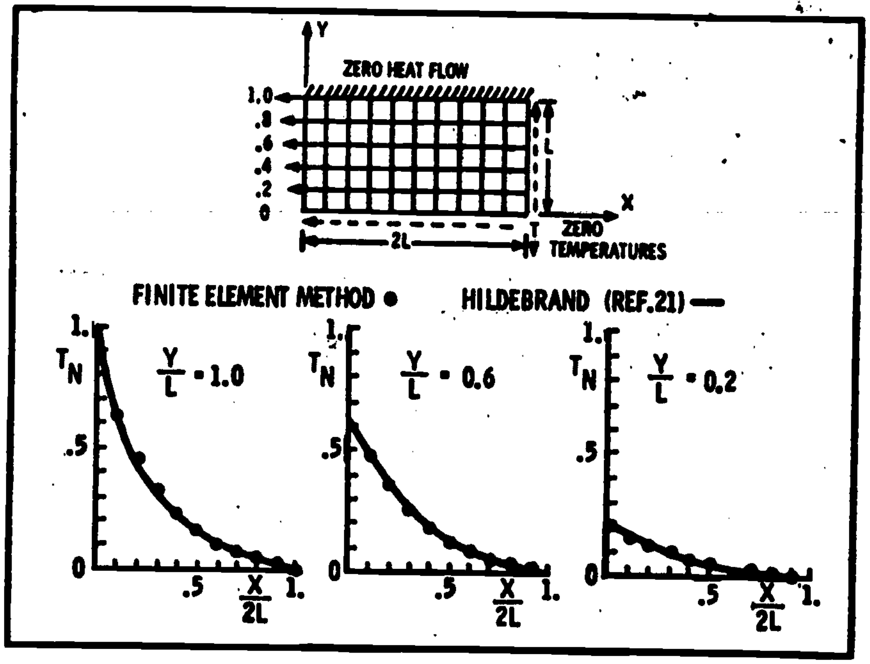


FIG. 15 PLATE CONDUCTION ANALYSIS

2. Cavity Analysis

The cavity illustrated in Figure 16 represents a problem in acoustics. The figure shows a two-dimensional view of a rectangular air cavity bounded on five sides by rigid surfaces and closed on the sixth side by a flexible panel. The problem is to determine the response of the combined panel and enclosed air volume to an impinging acoustical environment. The representation of the panel involves standard structural idealization procedures, and to represent the air system within the cavity a fluid finite element can be used.[19]

To test the quality of the representation, a finite element analysis was performed for an air system

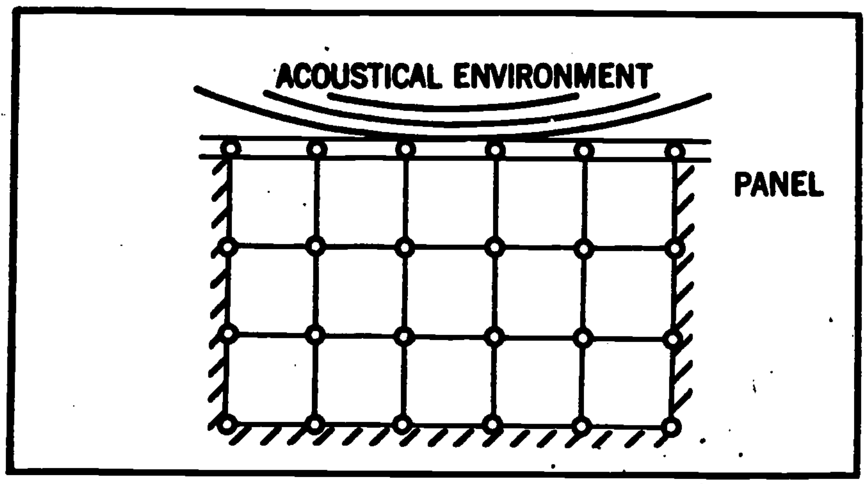
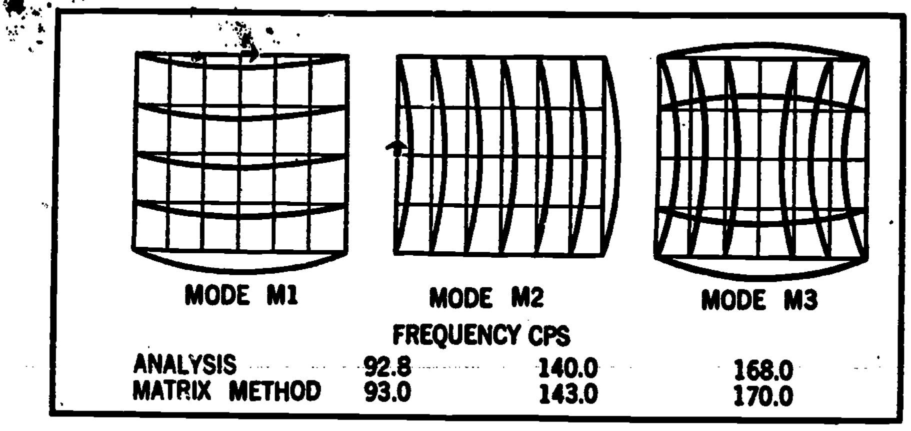


FIG. 16 CAVITY ANALYSIS USING FLUID ELEMENTS

surrounded entirely by rigid surfaces. Boundary conditions were such that motion normal to a surface was permitted. Submitting the assembled air system matrices to a vibration analysis yielded modes and frequences that correspond to standing waves in the cavity. In Figure 17, results for the frequencies of the finite element approach are compared with those obtained by classical analysis.[22] The two sets agree very closely and mode shapes were virtually identical for the two methods.



FREQUENCY CPS			
ANALYSIS	92.8	140.0	168.0
MATRIX METHOD	93.0	143.0	170.0

FIG. 17 CAVITY MODES AND FREQUENCIES

It can be seen that the ideas expressed here are capable of considerable extension, such as application to more complex configurations and to representations of fluids other than air.

REFERENCES

1. Argyris, J. H., "Energy Theorems and Structural Analysis," Butterworth, 1960.
2. Turner, M. J., et al., "Stiffness and Deflection Analysis of Complex Structures," Journal of the Aeronautical Sciences, September 1956.
3. Denke, P. H., et al., "A Computerized Static and Dynamic Aircraft Structural Analysis System," S. A. E. SP-264, 1964.
4. Gallagher, R. H., "A Correlation Study of Methods of Matrix Structural Analysis," A. G. A. R. D. 14th Meeting, Paris, 1962.
5. Klein, B., "A Simple Method of Matrix Structural Analysis," Journal of the Aeronautical Sciences, January 1957.
6. Irons, B. M. and Draper, K. J., "Lagrange Multiplier Techniques in Structural Analysis," A. I. A. A. Journal, June 1965.
7. Robinson, J. and Regl, R. R., "Automated Matrix Analysis for General Plane Frames," Journal of the American Helicopter Society, October 1963.
8. Clough, R. W., "The Finite Element Method in Structural Mechanics," Chapter 8, "Stress Analysis," Editors Ziekiewicz, O. C. and Hollister, G. S., Wiley, 1965.
9. Zienkiewicz, O. C., "Solution of Antisotropic Seepage by Finite Elements," A. S. C. E. Journal of the Engineering Mechanics Division, February 1966.
10. Fenves, S. J., "Structural Analysis by Networks, Matrices and Computers," A. S. C. E., Journal of the Structural Division, February 1966.
11. Bampton, M. C. C., "Matrix Structural Analysis—A Course in the Theory and Application of Matrix Methods of Structural Analysis to Airplane Structures," Vol. I, Boeing Document D6-2587 TN, 1964.
12. Kaufman, S. and Kaufman, J. J., "Molecular Vibrations by a Matrix Force Method," Martin-Baltimore Report ER 13536, June 1964.
13. Visser, W., "A Finite-Element Method for the Determination of Non-Stationary Temperature Distributions and Thermal Deformations," Conference on Matrix Methods in Structural Mechanics, Wright-Patterson A. F. B., October 1965.
14. Bampton, M. C. C., "The Matrix Temperature Method of Thermal Analysis," Boeing Document D6-4370 TN, 1964.
15. Argyris, J. H., "Modern Fuselage Analysis and the Elastic Aircraft," Butterworth, 1963.
16. Argyris, J. H., "Recent Advances in Matrix Methods of Structural Analysis," Pergamon Press, 1964.
17. Bendavid, D., "Application of the Direct Stiffness Method to Static and Dynamic Analysis of Pressurized Shells," Boeing Document D6-18113 TN, July 1965.
18. Kapur, K. K., "Buckling of Thin Plates Using Matrix Stiffness Method," Ph. D. Thesis, University of Washington, 1965.
19. Bampton, M. C. C., "The Stiffness and Mass Representation of Finite Fluid Spaces," Boeing Document D6-15163 TN (in preparation).
20. Archer, J. S., "Consistent Mass Matrix for Distributed Mass Systems," A. S. C. E., Journal of the Structural Division, August 1963.
21. Hildebrand, F. B., "Advanced Calculus for Engineers," Prentice-Hall, 1949.
22. Morse, P. M., "Sound and Vibration," McGraw-Hill, 1948.

FENVES: *I have two questions. First, how does the engineer describe an indeterminate structure consisting of 5,000 nodes or whatever the example was? Second, how can all this data be input with any reliability, considering the number of input cards that must be punched?*

BAMPTON: *That is a very good question and I know one or two engineers with bloodshot eyes who would sympathize with it. For one thing, the program that resulted in this analysis involving 5,000 nodes presumed a certain configuration so that the preparation of data was capable of being automated. Now this of course won't generally be the situation. But there was a certain prescribed configuration built into the program, namely, that particular problem happened to be a fuselage problem and it was known that it would consist of grids defined by stringer lines and* *frames, so this presented a lattice upon which the engineer could hang his data. This eased the problem which I think you implied by your question, namely, how can any engineer or team of engineers handle so much data.*

FENVES: *My next question is a little bit more fundamental and has something to do with the discussion we had earlier concerning the theoretical approach vs. the algorithmic approach. The procedure which you discussed was a very valid application of computers, that is, you take a continuous elliptic differential equation problem, chop it up into pieces so that you have only small pieces to concern yourself with, and then you use finite mathematics to combine these pieces together. Now you spend a tremendous amount of research on describing these particular elements. On the other hand, the technique, the algorithm, for interconnecting this very large number of elements so as to form something that can be computed, you dismissed in one sentence. You simply said, "We obviously don't use matrix algebra." Why is there this disparity of emphasis and of prestige? There appears to be a great amount of prestige associated with deriving new finite elements, but all the work involved in trying to put tcgether a couple of thousand finite elements, so that today's computers can get a solution rather than waiting till the seventh generation machines are big enough and fast enough, is tossed away as a coding trick.*

BAMPTON: *I don't know what is the basis for your assumption. I'm not tossing aside the problems involved in the system which is accepting the elements, and I didn't say that obviously we can't manipulate matrices in a tone which implied that this was a trivial matter. I just happened to have picked up a crutch, you might say, for discussion—namely, the finite element. There is no doubt about it; there is a lot of work going on in examining the system which incorporates these finite elements in order to handle them in the most efficient manner.*

SCHMIT: *I would like to ask if you could tell us anything about what effect, if any, the computer is having in your organization, or in organizations you are familiar with, on arriving at the system you had when you set about making this idealization; in other words, you described an aft end of a fuselage and you had something that you were ready to analyze. Certainly a great deal of preliminary design or some sort of effort went into bringing you to that point. Are computers having any impact on this sort of thing—the preliminary design phase? What I'm really concerned with is this: are we faced with the problem of optimizing what may not turn out to be a good idea, or attempting to get an efficient design based on a preliminary design which is not necessarily the best of alternatives? Is there any activity in using computers in this preliminary design stage in your organization?*

BAMPTON: *Yes, there is, and the activity is directed partially toward structural representation, with which I'm most involved, and also toward what might be called integrated analysis. That is, you have a structures program, a weight program, and aerodynamics program, etc., and you attempt to integrate these in order to get an efficient preliminary design. This has only been possible on a limited basis because you're faced with the task of preparing data. Thus, certain configurations have been assumed or built into the integrated dәsign program and they do not reflect a structure that can be sophisticated as our structural program can produce for us.*

CHASE: *I'd like to answer the question a little bit more. Actually there is a great deal of work going on in this area of the basic process. We have at Lockheed, for example, a bank of computer-system programs dedicated to the man-computer interaction that is being discussed today. In particular, one of the major areas of investigation involves the prelimiary design of aircraft in general, the surfaces, etc., in which structures of various sorts will be designed at some later stage. It is in its infancy. There's a great deal of work being put into it. Definition of surfaces is a complex problem within itself, but it is being conducted in industry. Some of the work is done at MIT under Coons in the computer-aided design program. But the whole concept of design in its many phases and many aspects is being investigated and we're whittling away at little pieces at a time. I think that you will see within a matter of two or three years quite a large accomplishment in which all these features will be put together in a somewhat more sophisticated package.*

PAYNTER: *I'd like to propose a question today which I'll have a chance to ask again tomorrow, but which is particularly appropriate to the present speaker. So far we have not yet come to grips with the problem of how can you evaluate whether a model is really any good or not. We saw here comparisons of a point-for-point type response between some test data and some model predictions. I'd like to raise a question in line with the comment just made in Fenves' second question: whether the following problem ought to be posed, that is, how does the presumed affect the decision process itself? In the context of* *many of your situations, you are not going to design plates with variable thicknesses at different points on the aircraft shell, so that, if you are restricted to using plates of uniform thickness, isn't it true that as long as the model predicts within one gauge thickness, you would accept it as an acceptable representation?*

BAMPTON: *I think the answer to that is in the affirmative, and I can only comment further that we have been investigating optimization programs using linear programming where such constraints as standard plate gauges have been entered into the programming problem.*

SAUNDERS: *I would like to inquire about the make-up of the teams who are doing this work. Are they predominantly of an engineering background, or mathematics, or computer people, or just who are they?*

BAMPTON: *It's a combination of engineers and mathematicians with the main emphasis on engineering. At Boeing, there is a distinct dividing line between mathematicians and engineers and they have to communicate across that line. Occasionally, mathematicians do cross that line, but the engineers are not allowed to do so. They cooperate pretty closely, and it is a combined effort that produces these kinds of formulations.*

APPLICATION OF COMPUTERS TO POWER SYSTEM ENGINEERING

G. W. STAGG AND A. H. PALMER
American Electric Power Co.,
New York, N. Y.

This paper will describe the application of computers in power system engineering and operation. It is essentially a case history of engineering computer applications in the American Electric Power Company.

The American Electric Power System provides electric service in a seven-state area. It has a generating capacity of over 8 million kilowatts, installed in 15 plants with 38 generating units. Power is distributed over 81,000 miles of transmission and distributing facilities. The System is connected to 21 neighboring power systems by 46 tie lines. In 1965 the peak load was slightly over 7 million kilowatts, of which approximately 99% was generated by steam units requiring 16 million tons of coal.

This particular case history demonstrates a more general operation than many others in that it involves electrical, civil, and mechanical engineering, as well as engineering information and project control applications. The achievements made in these areas will be described and some conclusions will be drawn concerning what the future portends.

Initially mention should be made of the background and resources which have been brought to bear on the problem. The word problem is used here in reference to provision to the practicing engineer of the computer capability of the digital computer, without requiring him to perform the task of program development. There exists a centralized group which is responsible for the development of computer applications.

For the past ten years, AEP has been requiring and training a staff of professional people. At present, the Engineering Analysis and Computer Division consists of forty engineers, analysts and programmers. In addition, a supporting group of ten people performs the computer operation, keypunching and computer services.

Two major groups operate with the Division. The Engineering Computer Applications Section is concerned with the development of programs for use in electrical, civil, and mechanical engineering activities. A prime responsibility lies in the evaluation and development of techniques suitable for computer implementation. The applications are generally large-scale programs which consider the entire problem area and are intended for system design and simulation.

The Engineering Information Processing Section develops programming systems for collection, processing, recording and analysis of the company's operation information in order to produce reports necessary for engineering control and evaluation, and also to satisfy regulatory requirements.

In each of these groups a distinction is made between engineer and programmer-analyst. The programmer, or analyst, generally provides a high level of mathematical training, whereas the engineer provides an extensive background in the analytical aspects of the problem area. There are forty baccalaureate degrees and thirteen advanced degrees among the forty people in the two Sections. It should be stressed here that there exist varied educational backgrounds within the division. Included are electrical, mechanical, civil, chemical and industrial engineers, mathematicians, physicists and business administration personnel.

The staff has grown partially by obtaining personnel from outside the organization, although of late some emphasis has been placed on recruiting from within, particularly in the Engineering Information Processing Section where experience within the organization is invaluable.

Mention has been made of the resources which have been used. An important facet has been the availability of computing at reasonable cost. Although the units of computer usage have increased considerably, the cost through the years has grown at a much slower rate.

A significant step in the continuing program of computer utilization for the solution of complex engineering problems occurred last year. On Nov. 1, 1965, an IBM/360 Model 40, specifically intended for engineering, was installed in the Service Corporation Offices in New York. This system is scheduled to be replaced by a Model 50 in December, 1966. It is expected that these machines will provide an impetus to additional expansion in the computer application area.

The efforts of this staff will be the concern of the remainder of this paper. Several broad subdivisions of application areas immediately arise. Although one of these is identified as design, all evidence a relationship to the design process to some extent. In the context which follows, design is to be considered the planned change of the AEP system. This is a continuing, complex, multi-step process which has as its aim the expansion and improvement of the system.

The several areas which will be indicated by example are:

1. Design.
2. Simulation.
3. Control.
4. Information processing.

DESIGN

Among the most promising engineering applications, from an economic viewpoint, are those in the design area. Programs of this type provide design assistance in connection with power-plant steel and concrete work and in location and design of transmission towers. In addition to material and design time-saving, the means are available for a more comprehensive analysis of each problem.

The power-plant design function is supported by a system of programs developed to aid the design engineer in achieving an economical design within time limitations associated with this type of work. The Structural Steel Framing Program is used in the design of plants of braced frame construction. In the design of a plant of this type, the flooring system, made up of rolled sections and welded plate girders, is designed somewhat independently of the bracing and column systems.

Data prepared for the flooring system is processed with or without specification of column requirements. As the floor design is developed, the reactions contributed to columns are automatically retained for column design. The normal sequence of events on the computer provides the designer with all flooring member sizings and, when required, column sizes.

An interesting extension to this program is the production of detail fabrication drawings. The information necessary for this step is all available within the design programs, and the technical and economic feasibility has been demonstrated of subsequently processing a magnetic tape file to produce drawings. The high-speed microfilm recorder displays information on a cathode-ray tube which is film-recorded to yield a drawing.

The Foundation Slab Analysis Program currently being used considers the mat to be a rigid body developing a planar soil-bearing pressure distribution, an assumption which is valid for the heavy slabs generally used in power-plant work.

The utility of this program lies in the possibility of evaluating a number of alternative designs and considering the vast number of load combinations. Although this is a fairly simple approach to the general problem of mat foundations, considerable work has been done in the investigation of a program to design the mat based on thin-plate bending theory for a plate on an elastic subgrade. A third program is also under development which considers the same foundation slab on piles.

In transmission line design, the computer enters the problem area at an early stage. Once a line plan is selected, field survey crews obtain terrain data. The field survey notebooks provide the data for the terrain model. It is hoped that eventually this data may be obtained even more directly by means of aerial photographs and subsequent automatic digitization.

In addition to the terrain data, it is necessary that constraints be specified for line crossings, conductor clearances and restrictions on tower positioning. The Transmission Tower Location and Selection Program uses a dynamic programming technique to achieve an optimum economic arrangement of tower locations and types. Several additional supporting programs available for the transmission area are sag and tension analyses of conductors, both for the design and field-stringing conditions.

Two salient factors should be noted. First, an integrated system of programs provides the capability of operating in the entire range of the problem area. Second, this system of programs, to be effective, must be integrated into the day-to-day work load.

A number of other special-purpose programs are available for use in civil engineering functions; for example, analysis of backwater curve, gravity dam, reservoir capacity and concrete stack.

SIMULATION

The planning and design of a large complex power system is virtually impossible without a computer. In the simulation of such a system it is necessary to develop a mathematical model which can be used for studying changes in the real system. By the use of this model, any number of system conditions can be simulated and their effects observed. These applications run the gamut from steady-state electrical network analysis to transient analysis.

A steady-state model of the electric power system is the load flow. This program evaluates station conditions and power flows and considers the voltage-regulating capability of generators, synchronous condensers and tap-changing-under-load transformers. In addition, desired energy interchange between systems is considered.

The system is described in terms of network connections, transmission line impedances, transformer impedances and capacitor data. For a given study year, loads, generation, voltage levels and interconnection data are given. The study results include a detailed report on the status of the system during the study period.

This program permits the planning engineer to consider a wide variety of cases. The effect of future load, generation, transmission lines, transformers, interconnections and voltage control equipment may be evaluated. Furthermore, the outage of various components can be simulated.

This type of program can be used to analyze vast systems. This particular version of the IBM/360 can analyze systems with as many as 2,000 stations and 4,000 lines, and, with random storage, systems many times this size. Until the general availability of digital computers this was not possible, nor were the solution techniques sufficiently well defined. Today, the design engineer has available the capacity to study in detail large interconnected systems, and thereby plan better future systems.

The Transient Stability Analysis program simulates behavior of the electrical network during switching operations and fault conditions. The differential equations describing the machines are solved numerically to determine the system conditions, following the specified operations. Of particular interest is the response of generators during the sequence of events. The aim in using this program is to provide data for use in the design of electrical facilities.

The Switching and Lightning Surge program also evaluates transient currents and voltages. The time scale of interest here is of a much shorter duration than in the transient stability analysis. In effect, a voltage wave is simulated in order to determine design criteria for lightning arresters and switching equipment. In addition, insulation requirements for electric facilities can be determined.

In the mechanical engineering area a program is available to evaluate a steam power-plant cycle. A one-line diagram defining the cycle is the starting point for a computer solution. The cycle is simulated by interrelating the various components of the cycle. A subprogram utilizing the characteristics of a component as defined by input data can simulate performance of a segment of the cycle. Subsequently, the remaining components are evaluated and eventually this iterative process converges. Results are provided for various points on the cycle. This program is used to design new cycles, evaluate current performance and to simulate modifications.

CONTROL (Operational)

The allocation of power-plant generation to effect optimum operating economy is an important problem in an electric utility system. The complexity of the problem for a large, integrated power system induced American Electric Power Company to install, in 1964, an automatic real-time control system to provide optimum operation of its power facilities. The facilities to provide on-line control consist of a 1,700-mile microwave system, analog and digital telemetering, analog equipment and a digital control computer.

The digital control system provides, on a real-time basis, economic loading schedules for 38 generating units and, in addition, performs power-system operating studies and logging functions on a time-shared basis. The control system has proved to be effective in allocating generation for optimum economy in AEP System operation as well as affording other savings.

The functions of the Digital Control System are:

1. Economic dispatch under normal conditions.
2. Study mode to permit investigation of energy interchange.
3. Logging.

INFORMATION PROCESSING

The logging function associated with the Canton control computer is a part of the AEP information collection system.

The MHW (megawatt hour) Data System establishes a history file from the hourly collection of generation, interchange and load data. Hourly and daily reports are produced automatically from this file and are retained in conjunction with regulatory requirements.

SUMMARY

In summary it should be indicated what plans exist for continuing the efforts described. The primary objective is to give to the practicing engineer caught up in the swirl of details a more effective path to computing capability. This is not to be construed as a path to the raw computer, as much as a channel of communications with the programs which are appropriate to the special problem. This entails a three-step process:

1. Development of integrated sets of computer applications associated with each problem area. These are well under way.
2. Development of a data base upon which these programs draw. Some inroads have been made in this area.
3. Development of a program to interrelate the problem program, the data base, the engineer and the computer. This presents a significant obstacle which at this point remains to be passed.

PAYNTER: *It looks to me as if you've outlined a system with a great many fragmented programs. You spoke of integrating these in a system of programs but how will Miller's master designer fit into your scheme of things?*

PALMER: *Well, I don't see that this is at all in contradiction to Miller's approach. I think this is* *what we are heading toward. Perhaps the path which we travel to get there is somewhat different from his, but I think basically we are coming to that same point.*

LICKLIDER: *Are there any parts of the spectrum in power distribution that are interference-free enough to consider sending signals over them, and, if so, are there any thoughts about sending computer signals over your power lines? And if not, are the problems legal or technical?*

PALMER: *In my experience the only time we became involved with something of that sort was several years ago when the Civil Defense people were thinking in terms of doing something of that sort. But, although I don't think much work has been done, I'm not really qualified to speak on this matter.*

COMMENT OFF MIKE: *Would you care to elaborate upon that?*

PALMER: *In AEP's case, I think this idea was pretty much replaced by the decision to go to a microwave system. But, in answer to the question, it is now being used by a number of companies for control systems, particularly in the far west where a power-line has been in use for twenty to thirty years.*

APPLICATIONS OF COMPUTERS IN HIGHWAY ENGINEERING PRACTICE

THEODORE F. MORF
Illinois Division of Highways
Springfield, Ill.

There is an old saying that has come down to us to the effect that "the cobbler's children go barefoot." As in the case of many other enduring expressions of traditional wisdom, this saying survives because it contains a large core of truth.

In a certain sense this saying has been true of the techniques of the engineering profession. Engineers, as a profession, are a very creative group—perhaps the most creative of all the professional groups that there are. Yet this creativity is directed outward toward the design of products and remarkably little change has been made within the engineer's own shop in the techniques tiiat he uses during the creative processes.

It always gives me the greatest pleasure to study the exhibit cases of the antique engineering, astronomical, and measuring intruments on display in the Adler Planetarium and the Museum of Science and Industry, here in Chicago, or in the Smithsonian Institution in Washington, D. C. Many, if not most, of these exhibited instruments are represented in a modern form in our present-day engineering offices.

If we seek differences between these antique instruments and their modern equivalents, we see less jewellike elegance and perhaps somewhat more utilitarian precision in the new ones. But, by and large, a modern engineer, draftsman, or surveyor could use without difficulty the instruments of his centuries-ago predecessor.

Indeed, I often think that Copernicus, who lived about 450 years ago, would have far less difficulty in accustoming himself to the instruments and techniques in a modern state highway department design office, than his wife—Mrs. Nicholas Kopernik—would have in preparing a batch of Polish Christmas cookies in my wife's modern kitchen.

Until recently, the most complex mathematical devices available to engineers were the desk adding machines and desk calculators. Far from being modern, we may be surprised to learn that completely workable models were constructed by Blaise Pascal, who died at the age of 39 in 1662, about 300 years ago. To add a personal note, Philip Matthew Hahn, a south German Evangelical pastor (and my great-great-great-grandfather) designed and built a calculating machine which was probably the prototype of the modern CURTA computer over 200 years ago.

Standing as it does in the mainstream of scientific and mathematical thought, the engineering profession has advanced on the break-throughs of the God-gifted geniuses of the past—Aristotle, Archimedes, Galileo, Napier, Newton; and more recently, Einstein, Fermi, and many others. We may all stand a little taller and hold our shoulders a little straighter when we acknowledge this heritage of genius, as it speaks to us today across the echoing chasms of the centuries.

While the bold advances in conceptualizations which we owe to these geniuses of the past and the present have normally led to advances in technology as we have come to comprehend their meanings, we may today be seeing something of an inversion of this normal process in the adoption of the electronic computer in modern uses.

The electronic computer, as it has been initially used, has been simply a far, far better mousetrap than we have had before. In its initial applications it has been like a $15 hamburger—an enormously advanced device. By its means we have done our conventional computations in an unconventionally rapid manner.

However, as we begin to appreciate the possibilities of the device, we find ourselves pressing against the limitations of our convention-bound techniques. We are indeed looking for new conceptual approaches to exploit, in presently unknown ways, the vast potentialities that we have discovered in the binary nature of an electrical pulse when used in combination with the rigors of yes-no logic. We are beginning to see, in real-time applications of computer control as well as in strictly mathematical processes, the intimations of a brave new world of conceptualization made possible by technical advances.

In the field of highway engineering it was my privilege to have been among the first to be exposed to the possibilities of the electronic computer. This was in December of the year 1955, and the place was the annual meeting of the American Association of State Highway Officials in New Orleans. Until then I had been dimly aware of the work going on at Harvard and a few other places. I had stood in the blinking, oven-like presence of Illiac at the University of Illinois. This was the "Tom Swift and His Magic Brain" period of popular journalism. As I remember, at this time a Univac was helping out on one of the popular television programs by finding, through mysterious mathematical means, a magical matching of matrimonial mates or something else equally preposterous.

Then at Miami in January, 1956, another manufacturer stimulated thinking among engineers by an exhibit at the American Road Builders Convention. My own first serious introduction to this equipment was in a small discussion group, following the 1956 Highway Research Board meeting, attended by only three or four highway engineers in the office of the late H. A. Radzikowski, at which his two assistants, S. E. Ridge and L. R. Schureman, described the results of their preliminary investigations into electronic computers during the past year.

The first electronic computer conference, under the auspices of the Electronics Committee of AASHO, was held in Chicago in early March, 1956, followed by meetings in Atlanta, Los Angeles, and Boston. Within about a year, nearly every highway department either had an electronic computer on the floor or on order. And within two years the various highway departments and consulting engineers in the highway field had over 1,000 programs in use applicable to the solution of highway engineering problems. In the years which have passed since then, the scope of applications has become so broad that inventories of numbers have become meaningless.

Without intending to venture into the jungle of superlatives as to which specialty, within the whole engineering profession, was the earliest, or the fastest, or the most efficient in applying the computer to engineering problems, I would say that the highway group as a whole has made a remarkable record in this respect. I would not be able to (nor if I could, would I wish to) bore you with a catalog of the highway engineering problems which have been aided by the electronic computer. These are all well known, and, if I could find a problem not yet within the range of analysis by computer, I would only be putting the spotlight of challenge on the next problem to be solved, in the words of the mountain climber, Mallory, "because it's there."

Certainly, the electronic computer, used as a calculating device, has profoundly affected highway engineering practice and all of those who are in highway engineering are aware of this change in varying degrees. The response to this change has by no means been uniform among individual engineers, as they show reactions ranging from tremendous enthusiasm through stolid acceptance and all the way to baleful skepticism; but not indifference. No thinking person can be indifferent when his mental cage is being so vigorously shaken.

How are electronic computers being used in highway departments? Of course, as in any other large business organization, there are fiscal, statistical and accounting uses of the equipment. However, since this conference is concerned with engineering education, I would like to confine my remarks to engineering uses rather than accounting uses, while always acknowledging their presence.

Apart from conventional data processing, the uses in technical applications are quite broad, ranging from information storage and retrieval through the fields of traffic assignments, traverse computations, structural computations, and soil mechanics if one might suppose there to be some kind of systematic or orderly continuum.

There is also a certain futility in describing how electronic computers are being used in highway engineering, either technically or organizationally, because changes are taking place so rapidly. Changes in the types of problems susceptible of solution, in the scope of programs, in the software which permits better operational scheduling of work, also more efficient, so-called small computers and more practicable remote operations on large computers are all factors which modulate the momentary advantages of any particular mode of use.

This is why a discussion of what any organization is doing must be understood to be a kind of amalgam of the outmoded methods and obsolete equipment that is actually in use, as modified by the intentions of use for the equipment on order, and liberally seasoned by ideas on techniques now only dimly perceived for the remote future.

Generally the use of electronic computation in highway engineering today is not in itself a system in the sense that data processors use the term, nor in the light of present knowledge is it likely to become, broadly, a system in the foreseeable future. Some parts might be termed thus, perhaps, but in the main, computations, whether done by pencil or by electronic computer, are only a part of a larger system of design.

Generally also, in the present state of the art, highway engineering problems are mainly of the analytical rather than of the design type. In the analytical situation, the designer sets certain conditions and the computer is programmed to calculate certain results or to optimize certain aspects. The designer reviews these results and, if he is not satisfied, he will change the variables and repeat the analysis.

This feature of a dialogue between the engineering designer and the computer often argues for the closest association between the two and against the big centralized computer with its theoretical efficiencies for data processing. Functionally, an electronic computer in an engineering analysis use might be compared simply with a tremendously powerful desk calculator available, if not immediately at the designer's elbow, at least in a menage where scheduling is not a formidable problem.

Another growing area of use of computers is in the real-time application of traffic surveillance and control. For some time highway engineers have thought of the traffic usage of an urban freeway as being analogous to any other continuously operating process. In a chemical plant, for instance, a sensing and feedback control of readings of pressure, temperature, and composition has enabled process automation through the operation of the servo values. Similarly a sensing of speed, volume, and density of traffic on a freeway will enable the optimization of vehicle throughput, or speed, or gross time of vehicles in the transportation corridor by controlling the traffic signals at the entrance and exit ramps of the freeway.

We now have the hardware and the knowledge for traffic controls of this kind and are, in fact, doing it to some extent on the freeways in the Chicago area, and in a somewhat different manner in Detroit, in river tunnels in New York, in Seattle, and in other places.

In Baltimore, New York, Toronto, and elsewhere, entire grids of traffic signals on surface streets are controlled by electronic computers on a real-time basis.

Further uses of process-control computers are seen in continuous sensing in the production of concrete, asphalt, aggregate, and other materials, but here the problem is less one of control than of knowing how to sense the variables that it is desirable to control.

What do we expect from the engineering schools in educating the engineers whom we will be hiring as graduates this summer and from now on? At the bachelor's level, not too much is expected beyond a general familiarization. What I am saying is that, given a fixed number of classroom hours, we would rather have the schools making our prospective employees better engineers than poor computer programmers.

At this level there should be several points of emphasis that are more important than instructions in how to manage a problem in any particular computer language:

1. First and foremost is the appreciation of the rigors of binary logic as applied to the solution of a problem. Every day we can see an increase in the evaporation of the velvet fog of "engineering judgement" and we are applying informed and discreet choices in more and more situations.
2. A disenchantment with the magic of ignorance concerning these fabulous devices and the acknowledgement that the limitations on computers are mainly the limitations of our God-given ability as humans to understand and analyze a problem or process.
3. Instruction in a pedagogical climate which acknowledges that yesterday's brave new world is already here and takes for granted the ever-widening application of computer uses.

As to specific computer instruction for practicing engineers of our staff, we have been conducting schools of instruction and would prefer to continue this practice. Our reason for this is that the instruction must be closely related to the specific computer or computers which are being used by our engineers, and the program languages are those which are applicable to our programs, and we want to continue having it this way.

Such schooling has been given to more than 500 of our engineers in both the central office and in the district field offices. A further advantage of having this instruction done by engineers of our computer staff is that it establishes their authority in the field and also aids in communications between those providing and those using the computer-center services.

Beyond the bachelor's level there would, of course, be specific instruction in one or more of the common program languages, and this is taken for granted. Here, however, the emphasis should be on the possibilities of the computer to lead the profession into higher levels of technical capability through better means of analysis and fewer approximations, or rule-of-thumb solutions.

As I reach the conclusion of my remarks I am struck by the paradox of some of the positions I have taken. I have written with deepest respect of the capabilities of the equipment and with some disdain as to their physical presence, or the language in which we must communicate with them. I am more appalled than impressed that some of them will generate seven acres of paper covered with printing in a few minutes. I conclude that I will have to explain this paradox to myself by seeing these devices as the man-made wings—wings unimportant in themselves— that will give added lift to the soaring spirit of man to enable him to move toward the fulfillment of those ends of creativity for the benefit of mankind for which he was made in the image of God.

LICKLIDER: *Two or three times there have been comments about the likelihood or unlikelihood of ever pulling together the myriad problem-oriented programs in a field into a system. This was something that was approached this morning by Miller and it seems as though we ought to get a little better agreement about the ground rules. I think there is at least one concept that doesn't suppose that you will ever pull sufficient* *number of programs together to form a system that will itself do design, if you mean a system of programs and computer hardware. But there is an alternative which says one should couple with the programs enough language implementation to make it easy for a person skilled in the art of the particular field to shape up, with the help of these already programmed resources, most of the software he needs to solve his particular problem. Now this coupling of the concept of language to the concept of a system of programs seems to me to be extremely basic and I'm thinking it would be important for this group to approach closer and closer to an understanding of what's possible there.*

MORF: *I think that there are about 12 reasons, and there might be 15 or 20, why this is a result that I see as being rather distant. I don't say that it is nonachievable, and in fact there are a number of efforts being made in this direction. One of these is being sponsored in part by the Bureau of Public Roads. It is called TIES: Total Engineering Integrated System. Another similar effort is being made at MIT, called ICES. There are problems here and they are very deep. One of the reasons is that we don't quite know how to score the ball game. Let's say, if you try to work a matrix on how to maximize the profits of a company, you have a rather simple objective. But what are we trying to do in the highway field? Does anybody know what the objective of highway administration or a road system is?*

MILLER: *We had some exposure, to TIES a year or so ago when it was first being formulated as an ideal. Our efforts are now devoted entirely to ICES, and this is different from TIES as is shown by the absence of the word Total. The ICES system, by decision, makes no pretense to be a total system. But it makes at least a start towards the goal of integrating what in the highway field, as an example, have been literally hundreds of piecemeal programs. And it is one single system, a whole set of languages or commands. The term integrated means that these sets of commands, or languages, are so implemented that a bridge designer or roadway designer, a geometrics designer or foundation designer can all cross-reference a common data base; meaning that any one designer who is concerned with the location of a highway can, via commands, carry out a locations study which sets up files which can be cross-referenced by another man doing, let's say, geometric design of an interchange. His files in turn can be cross-referenced and utilized by the man designing the bridge, and so on. There is a whole set of commands for the various technical disciplines which are integrated in the sense that one can transfer data between these sub-systems and carry out in one continuous process, using one master set of files, a rather complete design job. I agree with you entirely that this matter of the objectivity, the criteria, or what it is you're trying to do, is still open. The first operational versions of ICES will rely almost entirely on the designer himself, to provide these criteria. He will have a very powerful design tool; how he will use it, though, is still going to be up to him.*

MORF: *I didn't by any means mean to say that this wasn't an eventuality. But I think that there are a lot of considerations that have to be thought of before we have something that is available in the real world.*

BAUMANN: *We have heard various things about making languages more useful for people. One of the things that people do very well is use language redundantly; computers do this not at all. The real thing Miller is talking about is integrating some of these special programs by having them done under one roof. What we really need to accomplish in terms of language is to somehow make it possible for redundant languages to become useful. An example I think we might well look to is in the field of electronics. If you take a new television set down to your repairman, who may never have seen a model of this kind before, he can locate right away the power supply, the high voltage section, the picture tube, etc. In other words, there is a lot more pattern in the electronics set-up than there is in computer programs as we now use them. We somehow need to put this pattern back into our programming language so someone else can recognize a standard element by just looking at it. Maybe we can go completely to a system where we can program computers topologically rather than by a listing. This seems to be one of the directions which will be very useful in integrating some of these separate items; that is, we'd be able to take a look at a specific computer program and see what inputs and so on are required for another routine.*

MORF: *If that was a question, the answer is yes.*

PAYNTER: *I would hope that this issue, which has been raised several times, is typical of some problems that we might come to grips with in the next few days. It is your feeling, going back to Licklider's charge to this group, that it is just a matter of time before we progress to a situation in which we can expect in our computer discourse the same type of freedom that we almost demand of each other if we are working together on a problem? Or do you sense that there may be* *some fundamental limitations to what we will ever be able to attain in the way of an integrated design procedure?*

MORF: *Well, I have answered a similar question in one way. Now I will go ahead and answer it in another. This is an organizational function. There are people who determine alignments, there are others who determine the tension in lines, or those who determine the voltage distribution, and so on, and these different functions are really in various compartments. Perhaps one of the restraints on a totally integrated system is the organization and bureaucracy of a large industrial organization. This again would argue that something must be changed in the way of organization if you expect to put the punch card in at the front door and have a final design come out at the other end. This would involve cutting across quite a large number of organizational lines. None of these are impossible but they're not the kind of things you're going to get right away and they do provide restraints, and they also provide an explanation of the reason that the solutions in the earlier phases are undertaken one part at a time.*

COMPUTERS AND ENGINEERING DESIGN IN INDUSTRY

D. E. Hart
Research Laboratories, General Motors
Corporation
Warren, Mich.

I was asked to summarize industry's needs for future engineering graduates. Quoting from the advance abstract, "Industry needs engineers who can approach new problem situations with intelligence, imagination, and confidence, and who can arrive at satisfactory solutions in a reasonable period of time."

That statement was true ten years ago.

It will still be true ten years from now.

The abstract also states that the computer is one of the many tools which the engineer should be able to use. I am sure you would all agree that that is a true statement, and, if that was all there was to the story, there would be no need for me to say anything further.

Of all the tools available to the engineer, however, the computer undoubtedly has the greatest potential for increasing his productivity as a designer. Yet only a small fraction of the engineers now in industry—and only a slightly larger fraction of the engineers now entering industry from the universities—are making use of computers in design.

This is the problem which we all face: How do we put the power of the computer into the hands of the engineer so that he can make effective use of it as a design tool? And, from industry's standpoint, how do we get the computer into the main stream of industrial design activity?

What do I mean by "the main stream of design?" In the automobile industry, it is the design activity which starts with a blank sheet of paper and ends up with finished automobiles coming off an assembly line three years later.

Who works in this "main stream?" Included among the workers are artists and clay modelers in Styling, but the largest number are engineering designers in hundreds of drafting rooms throughout the Corporation who spend millions of man-hours each year designing GM's principal product—the automobile.

It's pretty obvious why these designers have not been able to use computers—the man and the machine don't speak each other's language. The language of computers is numbers, while the language of the designer is graphics—lines on a drawing. This language problem will be discussed in more detail later.

Let me elaborate on the concept of the "main stream" of design. In Figure 1, the main design stream is depicted as an arrow pointing to the right.

Outside of the main stream is a function labelled Research and Development. It might also be called advanced design, or some similar name.

The R&D function develops new technology which is fed into the main stream after it has been thoroughly tested, e.g., Oldsmobile's front-wheel-drive Toronado. The function is also available on a consulting and troubleshooting basis. The three bars in the main stream illustrate that it takes three years to complete the design of a new car, but that a new design cycle must be initiated every year.

This might be referred to as a steady-state design environment in contrast with the space-age design environment where the object is to invent a way to do something which has never been done before, e.g., get to the moon and back.

My subsequent remarks will be primarily directed toward this steady-state design environment—which I believe is reasonably typical of much non-defense industry.

Where in this environment are computers *not* being used—and why?

In the R&D area there has been a rapid increase in computer use as an analysis tool, but it is not being used as much as it should be. There is a phenomenon here which might be referred to as the Threshold Problem: an engineer will use a computer only if it's easier than some other way. Even if a program exists as a black box which converts data sheets into printouts, the engineer must still fill out the data sheet, get it key punched, and then wait a few hours for the results to come back from the computer. As an extreme case, this would be a ridiculous way to solve a single quadratic equation; however, an engineer might need to solve several hundred quadratics in a month—one at a time. If a program doesn't exist, the engineer can wait in line for a programmer or try to learn (or recall) programming in a language like FORTRAN; however, FORTRAN is not good for the casual user since there are too many details that can be forgotten.

But something exciting is happening. I like to call it conversational computing. We recently installed a GE 265 computer system with remote teletype terminals which operates on a time-sharing basis over telephone lines using the Dartmouth-developed BASIC system.[1] BASIC is easy to learn, and if the engineer does forget the rules he is reminded of them at once, not four hours later. Terminals are presently installed in test cells, drafting rooms, and engineers' offices.

An important point is that the threshold has been significantly lowered—the computer can now be used quickly to solve even one quadratic.

Most important, a whole new group of engineers are now using computers—primarily because they can remain involved in the solution of their problems on a minute-by-minute basis. I conclude that it's got to be good if an engineer will put up with a teletype terminal as a means for communicating with a computer.

Speaking of involvement, I recently received the Proceedings of the Third Conference on Engineering Design Education on "Authentic Involvement in Interdisciplinary Design."[2] This was a report on a series of workshops in which faculty members attemped to guide their own students through unstructured design projects. In this report, I was unable to find reference to the use of computers on any of the design projects.

My conclusion is that we indeed have not succeeded in learning how to put the digital computer into the hands of the engineer as a design tool.

Referring back to the main stream of design, you will recall that designers now working on drafting boards are unable to use computers because they don't speak the same language. It has become apparent that the solution is not to teach the designer the language of computers, but to teach the computer the language of the designer.

I would like to point out that two designers often engage in a "conversation" about a design problem. During this conversation, they make reference to drawings, handbooks, and models which have been constructed to help visualize the problem. They make sketches to help express their ideas.

Such a conversation is essential if the job is to be completed on schedule—it couldn't be done if they had to send letters back and forth to each other. I'm emphasizing this point again because most problems are solved on a computer just that way—by sending it letters. Three or four hours after asking the computer a question, the answer comes back, and if the question wasn't asked exactly right, the answer may be only, "WHAT?"

There is another aspect of the design process: Design can be said to be about ten percent inspiration and ninety percent perspiration. But the planning and the doing—or in this case, the designing and the drafting—are so tightly interwoven that the designer has had to be his own draftsman—he has to see what one line looks like before he can decide how to draw the next one. In a sense, he engages in a sort of conversation with his drafting board.

The conclusion which we come to is this: "If the computer is going to help a designer, he has to be able to use it himself," and to do this we must make the computer appear to the designer to be a kind of high-speed mechanical draftsman which he can control on a minute-by-minute basis as he now controls his drafting board.

In this way, the computer can do much of the routine drafting, and the engineer will have more time to dream up the real design which only a man, and not a machine, can do.

In 1960, after two years of study and experimentation, the GM Research Laboratories initiated a major research and development project whose purpose was to learn how to put the power of the digital computer directly into the hands of these designers, and ultimately to move the computer into the main stream of GM's automobile design.

This project is known as Design Augmented by Computers—DAC-I.[3] Let me stress Design *Augmented* by Computers—not design *by* computers. This type of work is often referred to as computer-aided design, a term originally used by MIT.

A key element in the DAC-I system is a graphic console which is attached directly to the Research 7094 computer (Figure 2). It makes possible conversation between the designer and the computer in the language of the designer which is, of course, graphics.

This graphic console is a one-of-a-kind unit which was designed and built to our specifications by IBM.

This computer can display drawings to the designer on the cathode-ray tube. He, in turn, can give directions to the computer by pointing at items in the display with the special pencil or by means of the keyboard.

Inside the computer, and available on a moment's notice, are stored his drawings and his drafting

[1] P. T. Shannon, "Impact of Time-Sharing Computers on Education in Engineering Design," Conference on the Impact of Computers on Education in Engineering Design, April 21-23, 1966. Sponsored by the Commission on Engineering Education.

[2] Proceedings, Third Conference on Engineering Design Education, "Authentic Involvement in Interdisciplinary Design," July 12-13, 1965: Carnegie Institute of Technology.

[3] E. L. Jacks, "A Laboratory for the Study of Graphical Man-Machine Communication," Proceedings, 1964 Fall Joint Computer Conference, The American Federation of Information Processing Societies.

tools—the computer equivalents of the T-square, triangle, ruler, compass, french curves, etc.

All of the many computer programs—the software—which make the DAC-I system work were developed by the GM Research Computer Technology Department.

The DAC-I system went into operation in early 1963, two and a half years after the project was started. This was a unique laboratory facility whose purpose, as I indicated before, was to learn how to put the power of the computer directly into the hands of the designer.

But one important element was missing—the designers who were supposed to be able to use DAC-I. Computer technicians can design computer programs, but we can't design automobiles.

Both Fisher Body and Styling agreed to assign designers to the DAC-I project. These were real designers, not computer people, and it wasn't long before they were making significant contributions based on their experience and knowledge of design. During their first year using DAC-I, they were able to design a complete trunk lid, both outer panel and inner panel, on an experimental basis. Figure 3 shows the results of their efforts. This drawing was recorded directly on film by the computer using the Image Processor unit of the DAC-I system.

In a written paper, it is not possible to describe the dynamic nature of the conversational interaction between designer and computer using a graphic console. For those who are interested, a 3-minute movie on DAC-I is available on loan.[4]

Figures 4 and 5 are examples of the type of image which is displayed for the designer at the console. Figure 4 is a blow-up of a small portion of the inner panel of the trunk lid of a car. It shows the cutout hole and reinforcement where the courtesy light will be mounted.

When the designer studied this display, he saw that the subassembly was not properly located relative to the surrounding surfaces. In only a few minutes at the console he was able to supply all of the information necessary to redesign this subassembly 1½" to the left. The results are shown in Figure 5.

With experimental applications such as this, we have shown that it is perfectly possible to place the power of the computer into the hands of the engineer so that he can use it as a design tool without becoming a computer programmer. I should point out that this was no small job—the DAC-I software runs into well over a million computer instructions. But with the DAC-I laboratory we have demonstrated how to make the computer available to a large group of engineers who previously had not been able to use it.

Perhaps there is a parallel between what I have just described and the field of engineering education.

If we change only a few words, Figure 1 pretty well characterizes the educational process. The arrow now becomes the main stream of education; the products are engineers. The university, too, could be described as a steady-state environment; it takes four years to produce an engineer, but a new crop of students enters each year. Most university computer use has also been in the R&D area, and so far it has had very little impact on the main stream of education.

I would like to suggest that the university's goal should be to learn how to place the power of the computer directly into the hands of the student and how to start moving the computer into the main stream of engineering education.

PAYNTER: *I gather that now we're ready to come back to Miller's master designer, because he is obviously not the person Hart was speaking of, certainly not the three-, four- or five-year product, although perhaps the six-, seven- or eight-year product. I got the distinct impression from Hart's picture of two men talking to each other over the drafting board that these people were not necessarily master designers, but simply competent, typical engineers. Would you care to suggest how you think schools might make this particular group of people aware of what they could do with computers?*

HART: *I think this comes back to the last topic, which was that part of the goal of the educational process is to impart to the student new concepts and ideas, and the goal of improved education, by moving the computer into the mainstream of engineering education, is to increase the rate of concepts per unit time that you're able to stick into the headbone of the student. My initial remarks were that we are interested in engineers who have intelligence, imagination and the ability to approach a new job with confidence (so far I haven't seen that the university imparts anything to the student but the intelligence and the rest of it he learns as he goes along). Part of the current attempt to get students involved in actual design problems during their university training is the wish to give them something more than tools to work with, and also some knowledge of how to apply these tools and what it means to apply these tools. I think this is very worthwhile. I would like to see the computer used more in this type of design workshop.*

[4] "Design Augmented by Computers: DAC-I," available from the General Motors Corporation, Public Relations Staff, Film Library, General Motors Building, Detroit, Michigan, 48202.

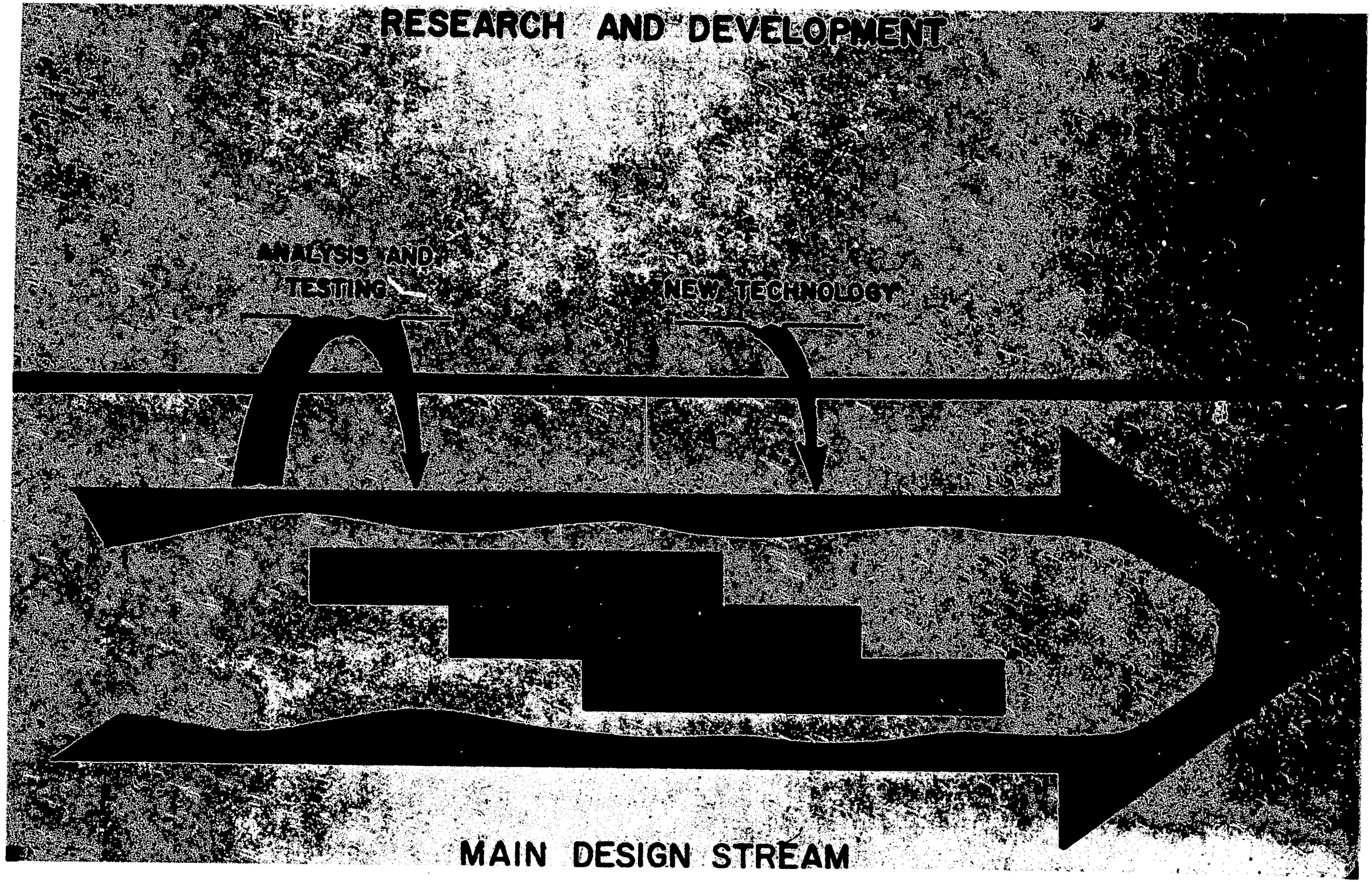
LOWE: *It appears to me that today we have sensed, in the minds of many people here, the feeling that there is a real barrier between the computer designer and the computer user. However, you said that the men who design the car body actually contributed to the design of the system, and that they came in as guinea pigs and in a very short time turned into contributors. I wonder if you could amplify this a little.*

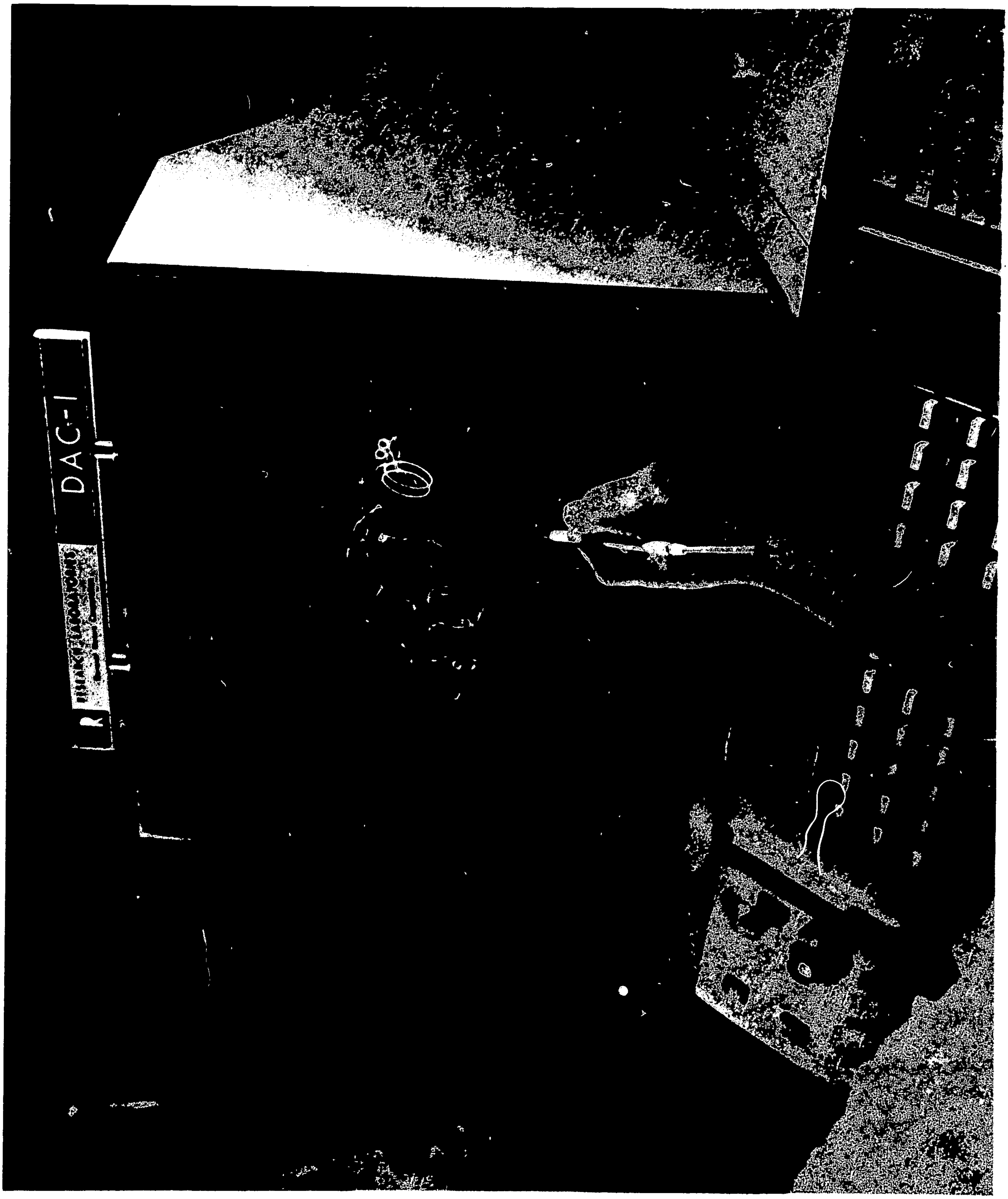
HART: *We could have spent ten years doing a systems study on how automobiles get designed. But we concluded that we weren't going to become smart enough fast enough to learn how to design automobiles so that we could design an automobile design system. So our goal and approach had to be entirely different. We observed one thing about the design process: whatever a man was designing, whether it was automobiles, spark plugs, or what have you, he used the same tools. He used a ruler, a french curve, a compass, and a straight edge. So our basic approach was to provide the designer with a set of reasonably primitive tools and means by which he could operate these tools in sequence in order to carry out a design process. As I indicated, we had to develop these tools in a kind of a vacuum, because, when we approached the designers and asked them what sort of things they would need if they were going to use a computer to help them with design, they just raised their eyebrows at us and suggested that we quietly retire to our laboratory and leave them alone to do their jobs. So we had to do the best we could to build what we thought would be an adequate set of tools to get started. And the the designers came in as guinea pigs and attempted to use these tools to solve real design problems on an experimental basis. They rapidly discovered that some of the tools we provided weren't good and some of the tools they needed we hadn't provided. But another phenomenon also took place: as they commenced to use the tools, they began to get some insight into the real capability which this new machine would provide for them. For example, if you work all day on a sheet of paper which is two-dimensional, you begin to think in terms of a two-dimensional world. We found that the designers who came in and worked with us did just that. They were working in a two-dimensional environment and it took them a while to adjust to the fact that, in working with the computer, they were really dealing with a three-dimensional medium, and they had to think in terms of working in three dimension. So their concepts of how they ought to do design changed as the tools changed. These two phenomena interplayed back and forth in developing the laboratory to the state where it is today.*

QUESTION: *Would not the same system, which could be the operational base upon which your engineers actually conduct design within General Motors, be a very powerful tool for teaching a new generation of designers within the engineering school? Suppose that we could afford to give the young men, before they came to you, two years of apprenticeship training under the guidance of some pretty good design people. Wouldn't the DAC system be a wonderful way to teach design in a university?*

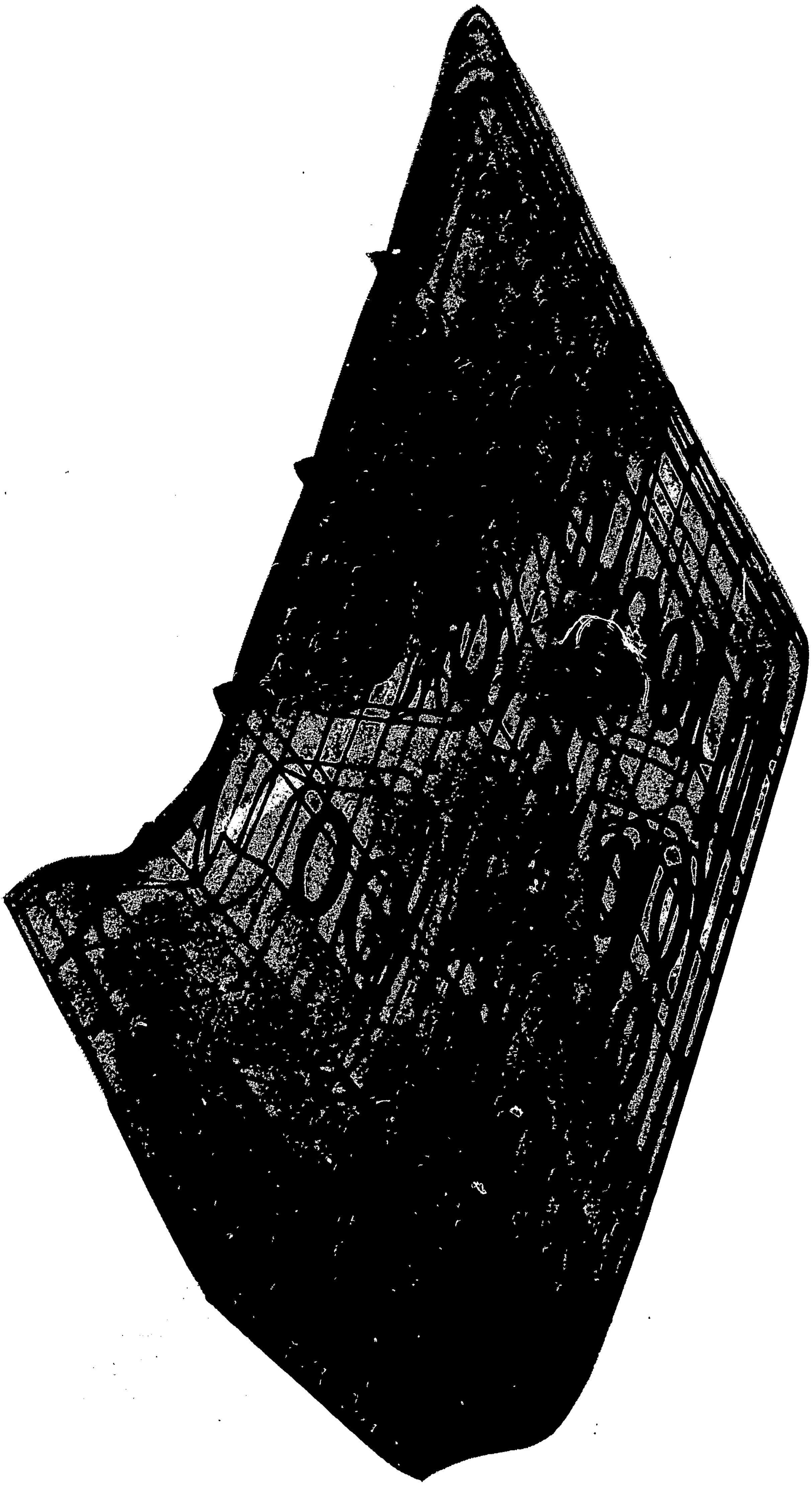
HART: *I think it would because it's a rather painless way to learn descriptive geometry and the visualization and interactions of things in space.*

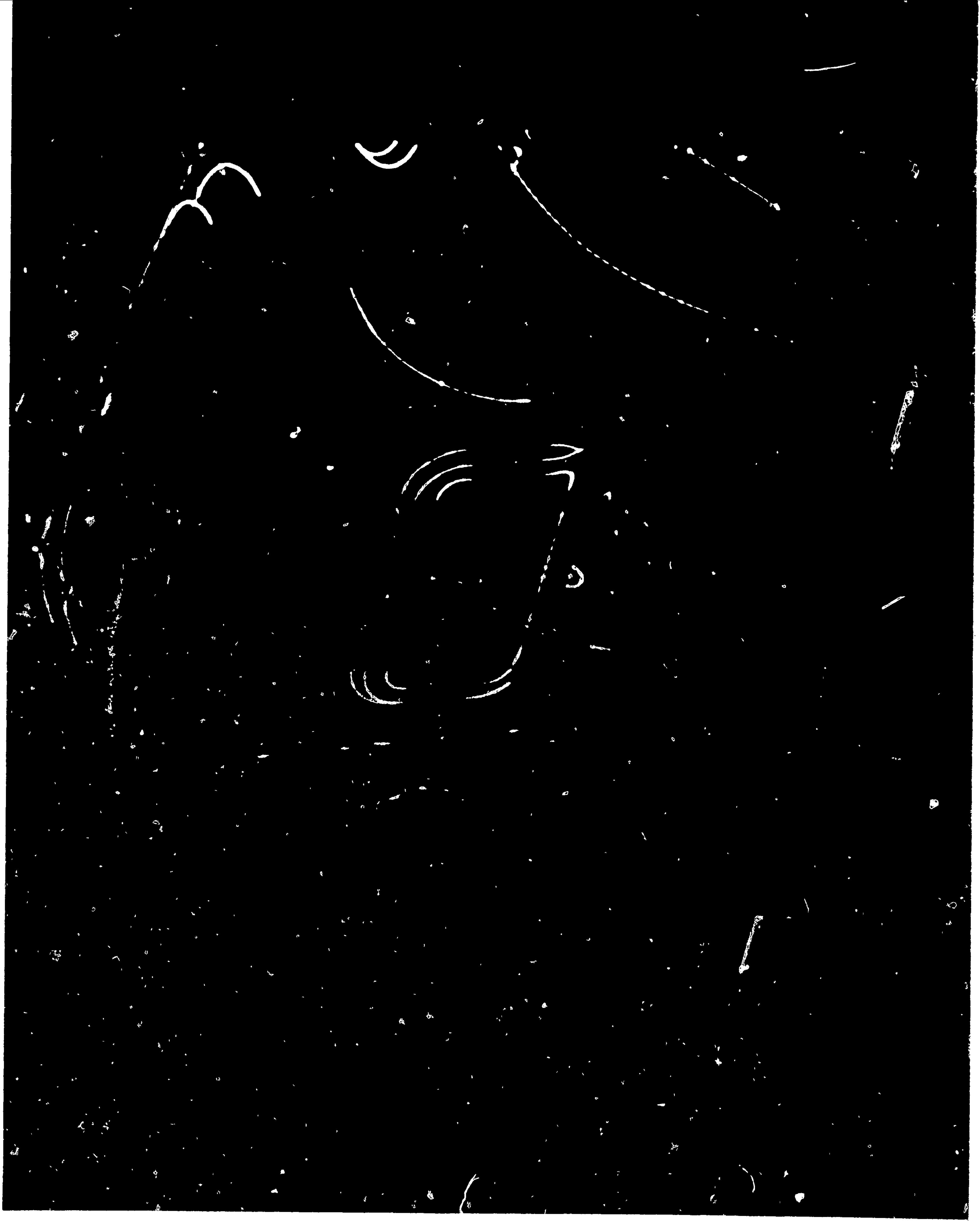
BRANIN: *I just want to make a brief summary of a couple of points which seem to have emerged from our session today. The question of accessibility to the computer has been brought out. It's really not accessibility to the computer per se, but to its facility. The other question was: how do we get engineers to use the computer? I think these points both have been very sharply highlighted by Hart's discussion.*

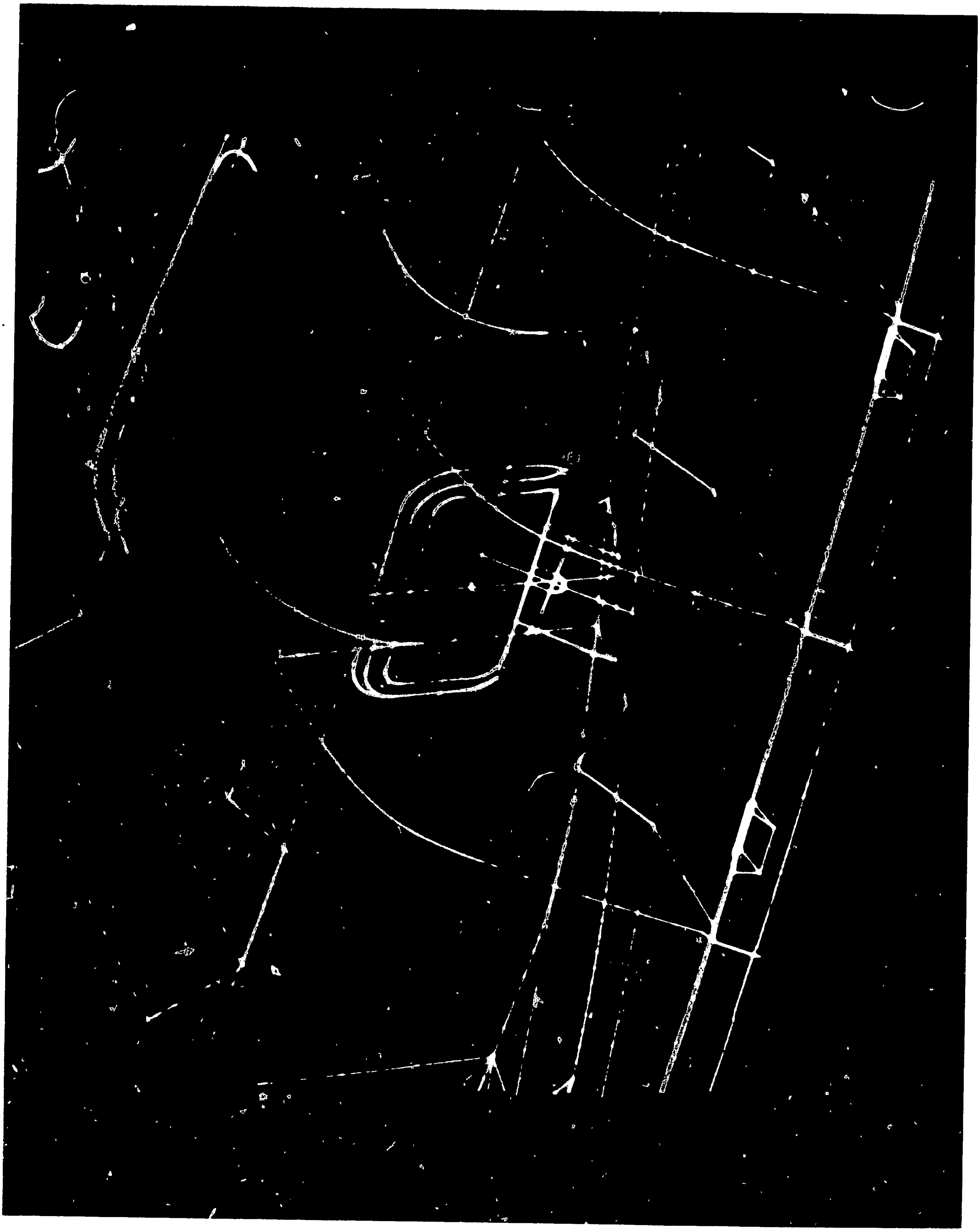

RESEARCH AND DEVELOPMENT
ANALYSIS AND TESTING
NEW TECHNOLOGY
MAIN DESIGN STREAM



DAC-1







COMPUTERS IN ENGINEERING DESIGN

Donald L. Katz
The University of Michigan
Ann Arbor, Mich.

At a meeting in 1962 of the Committee on Engineering Design of the Commission on Engineering Education, one task outlined for engineering educators was the development of the computer as a design tool. Following these discussions, a proposal was written to obtain support for the project which is being reviewed on this program. In the spring of 1964, a grant was received from the National Science Foundation for support.

ENGINEERING DESIGN

In the span of years encompassed by my contact with engineering education, engineering design has gone through three stages. The first stage was that of adopting the current technology along with modifications required by the new process under consideration. In 1930, as a student, I designed a petroleum separation plant utilizing information available on fired heaters, distillation columns, condensers, and treaters. Gradually there appeared a conflict between the developing engineering sciences and the art in form of technology. Accordingly, design problems of the second variety were developed. These problems required the creation of a process or system based on fundamental calculations for the various components using rigorous rate process equations, separation process relationships, and other quantitative material. Such designs became rather involved and tedious. It was necessary to make a good choice of the parameters in the first place to permit completion of the design problem within the time limits permitted in the educational environment.

With the advent of computer usage in the engineering sciences, it became rather clear that engineering design would soon be relegated to the computer—in good part because of the need for rapid calculations. It appeared that the universities should find a format by which design could be upgraded to include the computer techniques being adopted by industry. Also, the need for having design teachers conducting studies at the forefront of knowledge in this area was indicated. It was this view and the discussion with the Commission on Engineering Education's Committee on Design that prompted the preparation of the proposal for our project.

The project organized in the fall of 1964 at The University of Michigan was modeled after an earlier Ford Foundation project.[1] It had as major objectives: (1) the training of engineering design teachers in subjects related to computer-aided design, (2) the study of the role of the computer in engineering design education, and (3) the generation of a substantial number of completely documented computer-oriented engineering design problems.

In the fall of 1964, engineering design teachers with substantial computing experience were asked to apply for an intensive nine-week program to be held at the University of Michigan during the summer of 1965. An Advisory Committee of the Commission on Engineering Education[2] assisted in selecting from among the applicants twenty-nine engineering design teachers from twenty-three different engineering schools (list attached).

Five weeks of the summer program were devoted to formal presentations on computation, numerical mathematics, modeling and simulation, mathematical optimization techniques, problem-oriented languages, man-machine interaction, economics, and industrial design practice. During the remaining four weeks, each participating professor solved and documented one or more individual design problems appropriate for inclusion in an engineering design course in his own field.

During the fall of 1965, the project staff and one summer-program participant from each of five engineering disciplines edited or revised solutions to the participants' design problems and prepared material for the final project report. This report consists of six volumes, one summarizing project activities and five oriented toward design in individual engineering disciplines as follows:

Volume I.	Summary
Volume II.	Chemical Engineering
Volume III.	Civil Engineering
Volume IV.	Electrical Engineering
Volume V.	Industrial Engineering
Volume VI.	Mechanical Engineering

Volume I includes an historical description of the project, details of the summer program, recommendations, review papers on the subjects covered during the summer program, and abstracts of forty-three of the design problems prepared by the

[1] Project on Use of Computers in Engineering Education, University of Michigan (1959-63).

[2] N. M. Newmark, Chairman (University of Michigan)
S. E. Elmaghraby (Yale University)
S. J. Fenves (University of Illinois)
D. E. Hart (General Motors Corporation)
A. Schultz (Cornell University)

summer program participants. Each of the five disciplinary reports includes a brief statement on the role of computer-aided design in the field and several completely documented computer problems. These reports result from the combined efforts of the project staff and the first author as principal editor. The example problems in some cases were modified after correspondence between editor and problem author.

Some general conclusions and recommendations of the project staff are:

1. Engineering schools should continue to develop introductory computing courses to provide every engineering student with an adequate background in digital computation and instruction in at least one procedure-oriented programming language.
2. Computing work should be incorporated into the required upper-level analysis (engineering science) courses; before graduation, each student should solve many engineering problems of graded difficulty using a computer.
3. If the introductory programming course is taught early in the curriculum, as usually occurs and is recommended, then there should be a later treatment of more sophisticated mathematical topics available to engineering undergraduates. The student will need:
   a. More advanced mathematics, in particular numerical mathematics and mathematical statistics;
   b. more careful instruction in the techniques of mathematical model building and simulation;
   c. an understanding of the most important of the mathematical optimization techniques;
   d. exposure to a variety of computing subjects, including special purpose computing languages and the manipulation of graphical information.
4. The mathematical tools (numerical mathematics, mathematical model building, optimization techniques, etc.) needed for successful integration of computer work into engineering design courses are basically the same for all engineering disciplines, although there are some differences in emphasis among the engineering specialties. Therefore, it is not unreasonable to teach such material in an interdisciplinary course. At the same time, it should be recognized that problems are most meaningful when taken from the area of interest of the student. Hence, problems used in an interdisciplinary course would have to be chosen very carefully.
5. If we expect students to use the computer to its full potential for engineering design work and wish to assign rather sophisticated problems, special purpose problem-oriented languages should be used when available. If this is not practicable, then at the very minimum a substantial library of useful design subroutines should be available.
6. Design problems usually have an infinite number of feasible solutions, and experience and intuition play a significant role in the solution process. Engineering design procedures are often very difficult to automate completely. In order to use the computer most creatively, the designer must be able to converse with the computer in a virtually conversational way. Turnaround times of days, hours, or even minutes are not generally acceptable. The interaction must be effectively instantaneous.
7. Time-shared computers will have a great impact on the place of computers in engineering education in general. The student will be able to debug his program and solve a problem at one sitting. He should be able to solve reasonably difficult problems between class meetings. In addition, it will be practical for the engineering teacher to bring an on-line console, perhaps with graphical input and output, directly into the classroom. He will be able to display solutions for all to see as the computation is being done. Engineering faculty should be prepared to teach the use of such approaches within the next two or three years.
8. With the introduction of time-shared hardware and the development of problem- or design-oriented software, the student designer for the first time will have the opportunity to investigate a large number of feasible solutions to his design problem. He will be able to gain some facets of engineering experience which cannot now be acquired outside the industrial environment. Hence, the next generation of computing systems should enable the student to improve his intuition about and understanding of the nature of good design, which is, after all, the raison d'être of engineering design courses.
9. For financial reasons, it may not be possible or practical for a small school to acquire a large time-shared computing system for its exclusive use. The best approach for many schools will be to acquire remote terminals connected to a central processor shared with other university or industrial users. Even the larger schools will find the total cost of operation, though smaller on a unit computa-

tion basis, to be substantially larger than for the present generation of computing equipment. A very large increase in demands for computing services is almost certain to take place over the next few years. This will probably mean commitment of a growing share of the university's budget to support the computer and its services. University administrators should be planning now for support of these activities.

10. The key to successful integration of computers into engineering design course work is the knowledgeable and interested teacher. A considerable educational effort will be required to train our engineering design teachers to use the powerful new computing systems and languages.

Copies of the summary and individual disciplinary reports are being distributed to appropriate departments at all accredited engineering schools. Hardbound copies of the six-volume report are being sent to engineering libraries. Copies of individual report volumes will be furnished without charge to full time engineering faculty members. A limited number are available for sale to industrial organizations from the Publications Distribution Service of The University of Michigan.

During the 1966-67 school year, the project staff expects to conduct several two-day "information" sessions to stimulate interest in introducing computer work into engineering design courses. Representative engineering design teachers will be invited to attend local meetings to be held in New York, Boston, Atlanta, Dallas, Denver, San Francisco, and Ann Arbor. In addition, an intensive two-week workshop will be conducted in Ann Arbor for about fifty engineering design teachers.

Staff and Lecturers on Project

Brice Carnahan, Associate Director in Charge
Warren Sieder, Graduate Assistant
Dale Rudd, University of Wisconsin
Robert Morris, Bell Laboratories, Inc.
R. S. Miles, Autonetics, Inc.
C. Christensen and R. H. Roth, Bell Laboratories, Inc.
Robert Norman, Chevron Research
R. Gellatly, Bell Aero Systems
Don Hart, General Motors Corp.
Paul Shannon, Dartmouth College

Participants on Projec^

CHEMICAL ENGINEERING

Donald R. Brutvan, Associate Professor of Chemical Engineering, State University of New York at Buffalo

Rodolphe L. Motard, Associate Professor of Chemical Engineering, University of Houston

Thomas J. Schriber, Assistant Professor of Mathematics, Eastern Michigan University

Donald B. Wilson, Assistant Professor of Chemical Engineering, New Mexico State University

John M. Woods, Associate Professor of Chemical Engineering, Worcester Polytechnic Institute

CIVIL ENGINEERING

Rodolfo J. Aguilar, Assistant Professor of Civil Engineering, Louisiana State University

[1] Richard T. Douty, Associate Professor of Civil Engineering, University of Missouri at Columbia

Ti Huang, Associate Professor of Civil Engineering, New Mexico State University

Albert D. M. Lewis, Associate Professor of Civil Engineering, Purdue University

Allan J. Malvick, Assistant Professor of Engineering Science, University of Notre Dame

Charles H. Schilling, Professor of Military Art and Engineering, United States Military Academy

Wolsey A. Semple, Assistant Professor of Civil Engineering, Howard University

Theodore G. Toridis, Assistant Professor of Civil Engineering, George Washington University

ELECTRICAL ENGINEERING

Ernest E. Erickson, Associate Professor of Electrical Engineering, Louisiana State University

L. Dale Harris, Acting Dean of Engineering, University of Utah

Jens J. Jonsson, Professor of Electrical Engineering, Brigham Young University

[1] E. Lawrence McMahon, Associate Professor of Electrical Engineering, University of Michigan

INDUSTRIAL ENGINEERING

Harry Benford, Professor of Naval Architecture and Marine Engineering, University of Michigan

[1] Norbert Hauser, Associate Professor of Industrial Engineering, New York University (now Professor of Industrial Engineering, Polytechnic Institute of Brooklyn)

Gonzalo Mitre-Salazar, Professor of Industrial Engineering, Monterrey Institute of Technology and Advanced Studies (Mexico)

Frank D. Vickers, Assistant Professor of Industrial Engineering, University of Florida

MECHANICAL ENGINEERING

Rollin C. Dix, Assistant Professor of Mechanical Engineering, Illinois Institute of Technology

[1] Also served as participants in fall term 1965 to assist with preparation of reports.

Charles R. Mischke, Professor of Mechanical Engineering, Iowa State University

James P. O'Leary, Assistant Professor of Engineering Graphics and Design, Tufts University

Chester A. Peyronnin, Jr., Professor of Mechanical Engineering, Tulane University

Charles A. Timko, Assistant Professor of Mechanical Engineering, University of Notre Dame

Paul F. Youngdahl, Associate Professor of Mechanical Engineering, University of Michigan

[1] John R. Zimmerman, Assistant Professor of Mechanical Engineering, Pennsylvania State University

[1] Also served as participant in fall term, 1965 to assist with preparation of reports.

PREPARING THE UNDERGRADUATE FOR COMPUTER-AIDED DESIGN

CHARLES R. MISCHKE
Iowa State University
Ames, Iowa

The principal effort in the preparation of the undergraduate for participation in computer-aided design is made before his last design course. The preparation by the student is not as involved, nor as extensive, as superficial examination might indicate. However, the work to be done by the faculty seeking to offer computer-aided design is different to an extent that is greater than is generally appreciated. The purpose of this paper is to state the problem and to offer a technique for solution.

DEFINITIONS

An engineer plans and/or controls (i. e., designs) the interaction between matter, energy, men and money in order to accomplish a specified purpose. Design is the essence of the engineering function, although there are many engineering functions that are not design.

To design is to specify precisely how a task is to be accomplished. To specify requires creativeness and understanding.[1] Clearly, design is not taught in school. It is learned, if, indeed, it is learned at all, in the field, with and under a master artisan. Schools can, however, teach attitudes, approaches, conceptual tools and an appreciation of the morphology of design.

The basis of understanding is usually analysis, which is itself based upon mathematical and empirical models and experience, and is a useful means of evaluating alternatives. Selection of the best alternative requires:

(1) ability to recognize a satisfactory solution;
(2) ability to assess the merit of alternatives and order them accordingly.

Synthesis means the generation of a solution and it is arrived at by understanding the problem. The degree of merit of the synthesis varies directly with the ability of the student to understand. That is, poor understanding generates only weak solutions, whereas complete understanding may enable the student to derive a solution of high merit at the first attempt.

Engineering advances in the field of analysis promote understanding (and therefore the elegance of the synthesis) until the ultimate form of synthesis is possible. Although this state of affairs is a goal of engineering, its very attainment relegates such synthesis to the status of routine.

The frontiers of engineering include the excitement (and demands) of design work in a domain wherein understanding is poor, but the need is great. Hence there is a high challenge in searching for meritorious solutions by better analysis techniques (one route) or by clever search techniques in the domain of satisfactory alternatives (another route).

PERSPECTIVES CONCERNING THE COMPUTER

The role of the computer is becoming more powerful as the ability to simulate systems on it is developed. The computer calculates with speed, precision and without fatigue. Are these abilities useful to the designer? Certainly.

Should ability to exploit these capabilities be taught to undergraduates in engineering? The rapid growth of computer utilization in industry over the last ten years indicates "perhaps," but the advent of time-sharing computer hardware can lead to the development of a computer utility, like an electrical power utility, which could place in every engineering office a remote access to the largest computers available. The use of large computers will become a common part of design life. The advent of time-sharing will demand that four-year engineering graduates have some experience working fruitfully with a computer and have a broad understanding of current useful techniques.

Let us consider wherein computer-aided design assists engineering education, and what skills are needed. The engineering designer often proceeds as follows:

(1) Defines his problem in engineering terms.
(2) Decides how he will recognize a satisfactory solution.
(3) Determines how he will recognize a better (or best) alternative.
(4) Generates alternatives.
(5) Through analysis, retains suitable alternatives.
(6) Chooses best alternative.
(7) Acts upon the decision in (6).

Since the computer can only converse in a mathematical language, the hovering of a computer in the background affects these seven steps as follows:

[1] Strangely enough, although the handmaidens of design are creativeness and understanding, engineering schools do not select students on creative potential, nor do they as a rule make earnest constructive efforts to polish or cull on creative bases.

(1) The engineer must eventually prepare a mathematical statement of his problem for the computer. The pressure to state his problem in mathematical engineering terms is a wholesome influence.

(2) A description of a satisfactory solution in mathematical terms is required by the computer; every constraint upon solutions should be involved here. The computer requirements pressure the designer to be orderly, explicit, free of ambiguity, quantitative and exact.

(3) A figure of merit (utility function, object function, etc.) is always required by engieers. It may be non-quantifiable, such as an experienced expert's value judgement (a tea-taster, for example). The computer cannot function as a tea-taster unless the information and experience is placed in mathematical form. Again the pressure is upon the designer to make his utility function mathematical. (It also prevents double-talk, such as "I made it as cheaply and reliably as possible," while offering a mediocre alternative.) A computer program would reveal explicitly the designer's utility function. Again, the pressure to quantify judgements is wholesome.

(4) In this generation of alternatives the computer works with a speed and accuracy with which no man can compete. In this step the economic contribution of the computer is strong.

(5) Analysis programming has a history as long as that of the computer, and again the speed and accuracy of the computer exceeds that of manual calculation.

(6) Using the computer for this step removes much of the subjective prejudice in decision-making ("I like a cam for this application rather than a four-bar linkage."). Disenchantment with computer decisions tells the engineer either that his programming of step (3) is open to improvement, or that his prejudices are showing.

(7) The computer even contributes to the implementation of the decision. If the engineer needs clearance from a senior before he can proceed, the complete computer program together with the recommendation will allow the senior to quickly inspect the merit function employed and consider the implications. The latitude available to the decision-maker is apparent. A very flat figure of merit surface in the domain of the extremum allows considerable variation in parameters at small cost to accommodate non-quantifiable constraints, or whims, or prejudices of the decision-maker.

If we concede that the computer can be helpful to the designer (and therefore to the student designer), what is the price of giving a suitable introduction to computer utilization in design to the four-year engineering student?

First, let us be practical. If an engineering problem had to be solved and programmed from scratch on each and every occasion, the computer would find little use in the design room. The overhead to plan and debug the computer program would be large, and only a small number of programs would become economically feasible. Any notion that a problem once successfully completed would be encountered again should be suppressed.

If the overhead of time and money associated with computer use is substantial, how can previously invested time and experience be salvaged so that the long-term cost per problem becomes reasonable? The answer to this question is the same, surprisingly, whether you are talking about a human problem-solver or a mechanical one. A human is trained to provide conditioned responses to very elementary situations. A designer functions because he is able to bring to bear upon a problem a whole bagful of conditioned responses, arranged and ordered by him through intellect, in such a way as to solve the complicated problem at hand. Just as the human is taught a repertoire of conditioned responses, so can the computer be instructed. It then becomes the role of the human to arrange the responses into a sequential strategy for a solution. The computer capability corresponding to the learning and remembering of conditioned responses to elementary problems is called the subroutine capability of the computer.

A design room (or an educational department) requires a library of subroutines, available for quick access in any specified order by the designer (student) wishing to use the computer for design. The details on how to assemble and how to document such a subroutine library are not germane to my subject, but the existence of such a library is presumed for the remainder of this paper.

Given a problem in analysis, all that remains for the computer user to do is to plan a strategy of calling library subroutines in a specified order to initiate computation which will produce the required analysis in numerical terms. Our subject here is design by computer. The previous procedure is necessary to computer-aided design, but it is not sufficient.

What can a computer do? It can add, subtract, multiply, divide and perform other concerted actions which we simply call calculation. The fundamental and critical activity in design is decision-making.

Can the ccmputer make decisions? The computer can take direction and make logical decisions of the following kinds:

a. It can take direction to the extent of transferring control to a given location in the logic flow diagram of a program, regardless of the circumstances (the unconditional GO TO statement).

b. It can make a quantitative comparison and, depending upon whether a mathematical expression is positive, negative, or zero, it will branch in its logic and undertake a different course of action for each possibility (the IF statement).

c. It can make a quantitative determination and undertake a number of different courses of action depending upon the value of a mathematical quantity (the computed GO TO statement).

d. It can perform a calculation or a routine of calculation iteratively for any specified number of times (the DO statement).

e. It can read and write alphanumeric information.

f. It can make decisions based on the truth or falsity of a statement and proceed on a logic path dependent upon that determination (the Boolean algebraic capability).

g. It will remember (store in memory and recall) quantitative and alphabetic information, and it will seem to remember how to do a specified task in which it has had prior experience (the SUBROUTINE capability).

h. It can draw, i. e., plot, two-dimensional figures, and (with some computers) draw on a cathode-ray tube to communicate with the designer (the graphical output capability).

We can see that the computer has some decision-making capabilities. To the extent that the designer can reduce his decisions to combinations of the computer's decision capabilities, then the computer can be trained to function as a capable apprentice. In some things, such as in speed and consistency, it will excel its master.

The analysis capability of the computer with a library of subroutines serves the designer in step (5) of the design process. Step (2) is usually suggested by the constraints unearthed in step (1). Step (3), which is central to design, is often suppressed, or provided by a judgement. The computer demands that this step be quantitative. The designer must provide a quantitative relationship whose magnitude represents the merit of a satisfactory alternative. As soon as this is done, the problem remaining is to search over the domain of the satisfactory alternatives, and seek the one with the highest merit (largest value of the merit function). This is a procedure that can be effected with the computer and involves its decision-making capabilities. Step (4) requires creativeness and the computer (a model of consistency) has no creative talent.

A TECHNIQUE FOR COMPUTER-AIDED DESIGN

We have mentioned the elements of computer-aided design. A technique for making it practical consists of the following steps:

a. The designer-student writes an EXECUTIVE PROGRAM to read in all necessary information and write the output documentation to his problem. This executive program calls the SEARCH subroutine.

b. The SEARCH subroutine is a library program that will crawl over a merit hyper-surface and discover the location of the largest ordinate to the accuracy specified in the executive program. The search program calls the MERIT subroutine.

c. The designer-student writes the MERIT subroutine which generates alternatives and, if they are satisfactory, calculates their merit. In so doing this routine calls DESIGN library subroutines.

d. The DESIGN library subroutines are analysis routines that have been written, checked out, found useful and are documented in a convenient way to facilitate their use. (These are largely prewritten routines occasionally augmented by the designer-student.)

In the beginning of an operation, as in a school seeking to initiate computer-aided design experience among its students, the design library will be modest and prewritten by the design faculty, but, as problems are solved and new routines added to the library, an impressive design capability can be amassed.

Now, given the design library and the search program, just what mathematical preparation must the students have in order to use the computer for design?

The composition of merit functions is technically simple, but conceptually can be very complicated. This is not a computer subject; it is a design subject. When lacking much sophistication, one can resort to the use of the simpler merit functions such as weight, cost, or geometrical attributes.

Searching procedures need not be learned to any great extent because the search program is provided.

The designer's programming problems will be principally those of solving algebraic, transcendental, differential and simultaneous equations, evaluating integrals and derivatives, curve fitting, and using some probabilistic ideas.

To a large extent most of the places where these techniques are useful and applicable are within the design subroutines themselves, which are provided principally by the design subroutine library. To some extent the methods are used by the designer in constructing his merit subroutine.

The University of Michigan Project produced 43 problems, solved and documented by individuals working without the benefit of a design subroutine library or a rather general search program. Twenty-nine of the problems utilized optimization (search) techniques and were of varying complexity, cutting across most of the engineering disciplines. Nevertheless, the list of optimization techniques and numerical methods is neither large in extent nor impressive in complexity.

Thus, an instructor can choose several design problems constituting the entire computer-aided design experience wherein the level of sophistication and involvement with specialized techniques is in keeping with a very modest preparation and ability among his students. Not much numerical analysis preparation is essential to show the student the potential of computer-aided design.

Although some fine problems can be solved with a modest executive routine, how can some preparation in mathematical methods suitable to computer implementation be given to the undergraduate student?

A SUBTLE PREPARATION

Where, oh, where, in an already overcrowded curriculum, can this material be introduced without displacing other fundamental studies?

This is a perplexing question. Perhaps it is time for the engineer to re-examine his view of mathematics and its use in the light of the computer's role, present and future. The place to teach the material needed for solving equations, integrals and derivatives is in the mathematical sequences required of all engineering students and, to a lesser extent, in every engineering course in the curriculum.

Let's be specific. Is the time approaching when it would be more useful to view the entire spectrum of mathematics germane in engineering from a numerical analysis viewpoint? Cannot differentiation be taught from a finite difference and numerical analysis viewpoint? Cannot infinite series be viewed as partial sums with compensating truncation errors? Cannot integration be viewed as a summing up of rectangular (or trapezoidal) areas? Can't sophomores study linear algebra, matrix ideas, and solution to simultaneous equations (algebraic, differential, partial) together with computer implementation of same? Under these circumstances students would not know a Bessel function when they saw one, but do they know one now?

In other words, cannot the basis for understanding this mathematical material be brought into coincidence with the basis of computer techniques? To do so would place a double-edged sword in the hand of the student. If the mathematical sequence of some twenty-odd semester credit hours taken by engineers is taught to a lesser extent from the traditional viewpoint and to a greater extent from a numerical analysis vantage point, the student pattern of thought would be much more computer oriented, even if the computer were never mentioned! (And his mathematics would be no less useful.)

Given a new mathematical sequence and a simple programming experience in a language such as FORTRAN in the first year, and given continued interaction with the computer in the mathematics courses, more experience with it thereafter is required. In order to keep alive the student's programming ability and his numerical analysis ideas, every engineering course should require at least one programming problem each semester. I am not suggesting the use of just any old program to keep a curriculum committee happy. The program problem should be selected in order to make the course better for its inclusion. Such opportunities abound.

Under this arrangement, engineering students reaching their senior year and first comprehensive design course (not necessarily their first design course. There is something to be said for a design course every year) will be able to use the computer as a genuine tool in their design problems.

Design is a broad word and the meaning of the words design problem here is one in which the totality of the student's prior experience is drawn upon, one that taxes their creative skills, and one that requires a search for optimality (say cost) or a trade-off optimum (say weight, cost and reliability).

A FRINGE BENEFIT

Probably the most important reason for introducing computer programming early in the curriculum has to do with the handmaidens of design—creativeness and understanding. Engineering students are attracted to engineering partly as a place to vent their creative abilities. The usual engineering curriculum stifles this natural enthusiasm by demanding "conform now—create later" until by the senior year they are largly sheep. (If my colleagues doubt this, try the following experiment: Ask a group of seniors and a group of freshmen for an "off the top of the head" solution to a broad problem. You'll be surprised where the creative answers originate.)

The writing of a computer program is a creative act. Since there are many, many ways to skin a cat, creativeness can flourish here, and there is little cost associated with it. Mistakes are not preserved in cast iron, intricate machining or non-functioning equipment. The merits of alternative approaches can be demonstrated quickly.

Another act of the designer which is submerged in engineering education is that of decision. Courage in decision-making comes from confidence in tools of decision and a prior history of success. A computer program is replete with decisions and the consequences follow immediately (within the turnaround time of the computer installation). What better opportunity to sharpen the tools of decision and build confidence than in this aspect of computer programming? Thus, a simple course in programming can have secondary products of exercising creative and decision-making skills.

The changes indicated here will only come about through the initiative of the engineering educational community, courage to experiment, hardheaded evaluation, and considerable effort whose only sure reward is the excitement of doing and contributing. The dangers inherent in such changes are the same as in any change. Perhaps the most important necessary (but not sufficient) item is whole-hearted faculty support of whatever plan is implemented. Nothing dulls a student's enthusiasm more than to have had one professor assure him something is important, only to have a subsequent professor show disinterest (or ignorance).

SUMMARY

The computer can perform the design function. An undergraduate student can be given a meaningful introduction to the computer's capability in design if:

1. He receives a programming course in his first year.
2. His mathematics sequence is enriched with ideas and approaches from a numerical analysis viewpoint, and the computer is used as a teaching aid.
3. Every engineering course requires one problem in which the computer is used.
4. The faculty develops a design subroutine library in advance.
5. The faculty develops an efficient search subroutine.
6. Consideration is given to formulating figures of merit.
7. A design experience is permitted in which the student writes executive programs and merit subroutines to solve design problems.

Once convinced of the potential of the computer as a design tool, the engineering graduate will be motivated to learn on his own all he can concerning methods and approaches useful in computer work.

QUESTION: *You keep talking about developing a computer-programming skill and then keeping it alive until you can use it. Why not develop it just before you need it?*

MISCHKE: *Ideally, I think, a man ought to be exposed to design from the freshman year all the way through. There's a lot to be said for a design course every year in the man's experience, and it can start out very trivially in terms of background required and give some ideas concerning optimization and simple problems (they might even be ones to show up the math department) but problems which are too hard to do by pushing a pencil.*

QUESTION: *Are you then equating the use of the computer and the use of the design education as being co-equals?*

MISCHKE: *No. The computer is a tool. I think I made it clear that cooking up a merit function and figuring out how to generate alternatives are design activities. The machine does the rest.*

QUESTION: *I wasn't disagreeing with you, but I was trying to see whether you were saying that the purpose of introducing the computer into the engineering curriculum at all was for its use in design courses.*

MISCHKE: *No, other people ought to use it, too. At my school the amount of use by the disciplines within M. E. is exceedingly small at the moment. We hope to jar them a little.*

CRANDALL: *I know in our particurlar department in which people take four or five courses, we give them five hours of work and three hours worth of credit. Do you realize what a burden this might put on a student as far as an academic load if every one of these classes required a computer program?*

MISCHKE: *One program per course per semester? It doesn't have to be very sophisticated to teach the ideas. In fact that's the challenge.*

(laughter)

CRANDALL: *No, I'm thinking that in our basic computer course we require approximately five or six problems in the one-quarter system. If you do this in every quarter from then on, they are in effect taking an additional course load as I see it.*

MISCHKE: *No comment.*

CARNAHAN: *I think the new computing systems are going to make quite a difference in what can be done in the way of assignments. That is, in the past it has taken a minimum of two weeks to do a problem with any complexity at all, but with new systems I think it will certainly not be unreasonable to ask a student to do a problem between Monday and Wednesday if he can sit down at the console and get six runs and debug his program.*

THE SIMULATION SPECTRUM

NORBERT HAUSER
Polytechnic Institute of Brooklyn
Brooklyn, N. Y.

This presentation is based on the Simulation Chapter of Reference (1), which has been distributed to Conference attendees. The following remarks confine themselves to a few aspects of model building and digital computer simulation.

A process or system model is of practical value only if (a) it adequately represents the process, and (b) it is amenable to some form of mathematical or logical analysis leading to a better understanding of the process.

Currently available methods to implement (b) frequently require violations of (a). When there is a conflict between realism desired and power and elegance of approach, the choice appears to be between seeking (1) the exact solution to an approximate problem, or (2) an approximate solution to the exact problem. The first choice (1) is perfectly acceptable if the resulting distortion does not significantly affect the problem's characteristics, or if the engineer is able to evaluate and counteract the distortion's consequences. The second one (2) on the other hand implies the ability to duplicate an entire system (or to operate on it directly). Even if this were feasible, the time and expense required to design and execute controlled experiments necessary to isolate relevant variables makes this approach a last resort. In most cases the engineer looks for a point somewhere between these extremes. He is quite satisfied with a formulation which has abstracted the system's essential[1] characteristics, permitting him to apply other than brute force methods.

Simulation is one of the available methods. It requires no assumptions of linearity, differentiability, or any other form of mathematically good behavior. A set of mathematical and/or logical statements describing the exact (in a deterministic or stochastic sense) relationships among all components is needed. The simulation then produces the overall effect of these iteractions. It is important to realize that simulation need not be considered solely an alternative to analytical methods. Many times both approaches may be used concurrently. Indeed, one of the cardinal rules of efficient simulation design is *not* to simulate those parts of a model which can be treated analytically.

MODEL VALIDATION

Since it is impossible to prove the validity of a given model, the best that can be hoped for is that it will withstand a series of tests, performing reasonably and producing reasonable results. In analytic models the danger arises from oversimplification introduced in order to make the model suitable to mathematical analysis. In simulation, however, the danger frequently is of the opposite kind. The desire for realism may cause some important features to be overwhelmed by a mass of realistic but trivial information. Assuming the primary purpose to be a gain of insight into a system's cause-and-effect relationships, it is essential that all material irrelevant to the particular investigation be left out. This introduces some circularity: if we knew exactly which variables are relevant, we probably would not need to simulate. Model construction thus becomes an iterative procedure consisting alternately of (a) selecting variables and specifying their effect on each other, and (b) running the model and observing its behavior. If this behavior is unsatisfactory, either some relevant variables were excluded, or the interactions were inadequately specified. When the behavior appears to be reasonable, the sensitivity of all variables and parameters should be explored. Besides leading to the possible elimination of some variables, this will indicate the precision required of parameter estimates and other data used.

Not until we have reached this point are we ready to collect data from the system, and to determine how much time and effort should be expended in improving the accuracy and precision of this data. The mistake of building the model around available data, or of collecting the data before constructing the model is frequently made. The previous discussion should make the improvidence of such an approach apparent.

EXPERIMENTAL DESIGN

Since simulation as such provides no algorithm leading to improvements in succesive runs, effective search methods (machine or man-machine) have to be incorporated. This topic, including an extensive bibliography, is discussed in Reference (1). As in all experimental techniques, thought must be given to designing a method which will yield maximum information for a given amount of computation. If random events are simulated, estimates of the precision of computed output statistics are needed.

[1] Essential with respect to a particular application. Thus a model designed to answer one set of questions about a system may be worthless when confronted with another set about the same system.

Alternately stated, how many runs are needed, or how long should a single run be, to achieve a given interval of precision at a stated level of confidence? To answer these questions, the simulator must have considerable knowledge of probability and statistics.

COST

The cost of programming, debugging, and running a simulation are usually high. Furthermore, it invariably requires more time—planning, testing, executing—than originally anticipated. The availability of simulation languages may reduce programming time and cost somewhat.

Simulation does offer some outstanding advantages, however. It can be used effectively in training and demonstration. Sometimes, valuable insight can be gained by simulating a system which could also be treated analytically. This insight may lead to a better analytic formulation. Like a good doctor, lawyer, or teacher, simulation may achieve its greatest success by making itself superfluous.

SIMULATION LANGUAGE

A host of specialized digital-computer simulation languages, intended to facilitate the construction of programs ranging from discrete queueing models to analog representations of differential equations, are currently in use, and are constantly being improved. The reader is referred to References (1) and (2).

References

(1) Katz, D. L., Carnahan, B., Douty, R., Hauser, N., McMahon E., Zimmerman, J., and Seider, W., *Computers in Engineering Design Education*, Volume 1, University of Michigan, 1966.

(2). *Bibliography on Simulation.* I.B.M. Publication No. 320-0924-0.

OPTIMIZATION METHODS — A REVIEW AND SOME EXAMPLE APPLICATIONS

BRICE CARNAHAN
University of Michigan
Ann Arbor, Mich.

INTRODUCTION

During the 9-week 1965 summer program of the Michigan Project on Computers and Mathematical Techniques in Engineering Design Education, several days were devoted to formal presentations on mathematical optimization methods. Principal emphasis was given to those optimum-seeking and mathematical-programming methods which are best suited for implementation on digital computers and which in addition do not require a mathematical background much beyond that of typical senior-level engineering students.

During the final four weeks of the summer program, the twenty-nine participating engineering design teachers formulated, solved on the computer (IBM 7090) and documented one or more design problems suitable for inclusion in their engineering design courses. Most of them attempted to use one of the mathematical optimization methods to produce a design that was "best" in some sense.

These problems have been published in the five discipline-oriented volumes of the final project report[1] which have been distributed for the first time at this conference. In addition to these volumes, a summary volume (Volume 1) will soon be available; it contains, among several others, a lengthy review paper on mathematical optimization techniques. The paper includes a bibliography of 65 pertinent references which should prove useful to those interested in introducing optimization methods into the design-course environment.

The present paper is a considerably less detailed review of some of the better known mathematical optimization techniques and is deliberately brief and incomplete. No doubt there are exceptions to some of the generalizations which appear. I hope that these remarks will give the reader the flavor of the subject and lead him to study the source materials for details and rigorous mathematical developments. Some example applications of the methods will be described briefly in the latter part of the paper; those interested in details should refer to the appropriate discipline-oriented report.

ENGINEERING DESIGN AND THE OPTIMIZATION PROBLEM

Virtually every engineering design evolves in an iterative or trial-and-error fashion. The design problem is often stated in an ambiguous and open-ended way. The designer's first task is to decide what the problem is, and what the requirements or specifications for any proposed solution are. The designer then generates a design concept, usually in the form of a rough configuration, for the object, system or process being designed. These phases of the design activity are most difficult and require the greatest ingenuity and engineering judgement. The designer next attempts to "model" the object, system or process. This model may assume a wide variety of forms, but those usually preferred are mathematical in character. The engineer may use many weapons from his arsenal of mathematical and scientific tools, from the most fundamental of nature's laws to the most empirical of data correlations, from the most rigorous analytical mathematical tools to the least formal cut-and-try methods, to assist in the analysis of the proposed design.

The mathematical model normally contains several (say n) parameters which we will call the "design" variables, $x_1, x_2, \ldots x_n$. To simplify the notation, we will use the notion of a design "vector," $\bar{x} = [x_1, x_2, \ldots x_n]$. We may view each design variable as one dimension in the "space" of the design variables and any particular set of values for the design variables (i. e., any particular design vector) as a "point" in this space. It may happen that certain equality constraints

$$g_1(\bar{x}) = 0$$
$$\vdots$$
$$g_r(\bar{x}) = 0$$

and/or inequality constraints

$$g_{r+1}(\bar{x}) \leq 0$$
$$\vdots$$
$$g_m(\bar{x}) \leq 0$$

are imposed on the space so that only points in a portion of the total space may even be considered as acceptable candidates for solution of the design problem. Such a region is termed the "feasible"

[1] Katz, D. L., Carnahan, B., Douty, R. T., Hauser, N., McMahon, E. L., Zimmermann, J. R. and Seider, W. D., *Computers in Engineering Design Education*, Volume I - Summary, Volume II-Chemical Engineering, Volume III-Civil Engineering, Volume IV - Electrical Engineering, Volume V - Industrial Engineering, Volume VI - Mechanical Engineering.

space; points in this region are termed feasible solutions or feasible design alternatives.

The equality constraints come from functional relationships among the n variables which must be strictly satisfied; these often follow from elementary conservation principles. If there is to be a design problem, i. e., if there is to be a choice in the values which at least some of the design variables may assume, then r must be smaller than n. The inequality constraints usually come from some specified design limitations (maximum permissible stress, minimum allowable temperature, etc.) There is no upper limit on the value of m.

Briefly stated, the optimization problem is to find that one design vector or point in the feasible solution space which corresponds to the "best" design. In order to locate the best feasible solution of the usually infinite number of feasible solutions using a mathematical optimization technique, it is essential that a meaningful, computable, quantitative, single-valued function of the design variables be formulated to serve as a measuring stick for comparison of feasible design alternatives. This function is usually called the "merit," "effectiveness," "criterion," "utility," or "objective" function.

One of the designer's principal responsibilities is to construct a suitable objective function to be maximized or minimized. The objective function might be quite simple and relatively easy to calculate (the total weight of a storage tank to hold a specified amount of some material), or quite complicated and difficult to calculate (the rate of return over a 20-year period for a proposed oil refinery, taking into account likely variations in markets, raw materials, taxes, etc.). The objective function may be very illusive due to the presence of conflicting or dimensionally incompatible subobjectives; for example, one might be asked to design a plant of minimum cost which at the same time minimizes air and water pollution and is aesthetically appealing. In such cases, it usually will not be possible to find a quantitative objective function and hence to automate the selection of better design alternatives using one of the mathematical optimization procedures. In what follows, I assume that a suitable objective function has been formulated by the designer.

The mathematical optimization methods can be classified in a variety of ways; one such classification is shown below in Table I.

TABLE I

Classification of Optimization Methods

1. Optimization methods for unconstrained functions of one variable.
2. Optimization methods for unconstrained functions of n variables $(n > 1)$.
3. Optimization methods for constrained objective functions.
4. Optimization methods which exploit system or information flow structure.

Optimization Methods for Unconstrained Functions of One Variable

Some optimization methods for unconstrained functions of one variable are listed in Table II. One criterion for locating a local extreme point of an unconstrained, continuous and differential objective function f(x) is that its first derivative f′(x) vanish. The traditional tool for solution of such problems is the differential calculus, but its applicability in design problems is severely limited by the complex and nonlinear nature of most objective functions.

TABLE II

Optimization Methods for Unconstrained Functions One Variable — f(x)

I. The Calculus

II. Numerical Mathematics
- A. Root-Finding Procedures for f′(x)
  1. Newton's Method
  2. False Position Methods
  3. Half-Interval (Bisection) Method
  4. Methods for Polynomial Functions
     - a) Graeffe's Method
     - b) Bernoulli's Method
     - c) Iterative Factorization Methods

III. Searching Schemes
- A. Simultaneous Methods
  1. Exhaustive Search
- B. Sequential Methods
  1. Dichotomous Search
  2. Equal Interval Multi-Point Searches
  3. Golden Section Search
  4. Fibonacci Search
  5. Lattice Search

Some tools from numerical mathematics may prove useful, particularly the numerical root-finding procedures listed in Table II. Newton's method requires the computation of the second derivative f″ (x) as well as the first derivative f′ (x). The false postion and half-interval methods require only that f′(x) be continuous and computable. Should the objective function be a polynomial, then one of the listed root-finding procedures for polynomials might also prove useful.

In contrast with these numerical methods, the searching schemes listed under parts IIIA and IIIB of Table II require only that the objective function f(x) be computable; for most of these procedures, neither differentiability nor continuity of the objective function is essential.

In each case, the optimization problem is to locate the global extreme value of an objective function f(x) on some arbitrary starting interval A≤x≤B, such that final optimal x value found by the algorithm and the true optimal x value lie within a specified "interval of uncertainty," i. e., an interval of specified length known to contain the extreme value. To simplify comparison of the methods, we assume that an arbitrary starting interval [A, B] has been mapped onto the unit interval [0, 1] by a suitable linear transformation. Hence, the initial interval of uncertainty for each method will have length 1. Other nomenclature used in common throughout this section is:

a = length of final interval of uncertainty

N = total number of evaluations of f(x) required to reduce the length of the interval of uncertainty from 1.0 to a.

n = the number of iterations for the sequential searching strategies (see below).

The one-dimensional searching schemes may be classified into two categories termed simultaneous and sequential. In the simultaneous schemes, all points at which the function is to be evaluated are selected a priori. One such approach termed "exhaustive search" is illustrated in Figure 1. Here the function, which need not be differentiable or even continuous in some cases, is evaluated at equally spaced intervals, a/2, assumed to be small enough to catch all pertinent features of the function. The total number of function evaluations required is

$$N = \frac{2}{a} - 1$$

Clearly, this is not a very effective approach for small a, particularly when the function is difficult to evaluate.

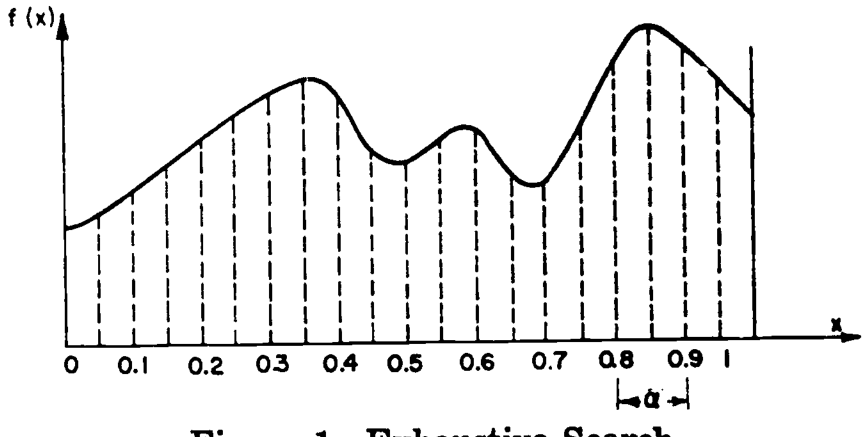


Figure 1. Exhaustive Search.

When the sequential methods listed in Table II are used, the arguments for which the objective function will be evaluated cannot be known a priori; instead, the sequence of argument values depends uuon the already observed values of f(x). All these methods require that f(x) exhibit unimodal character, i.e., exhibit just one local extreme point on the interval of interest as shown in Figure 2. The function need be neither differentiable nor continuous.

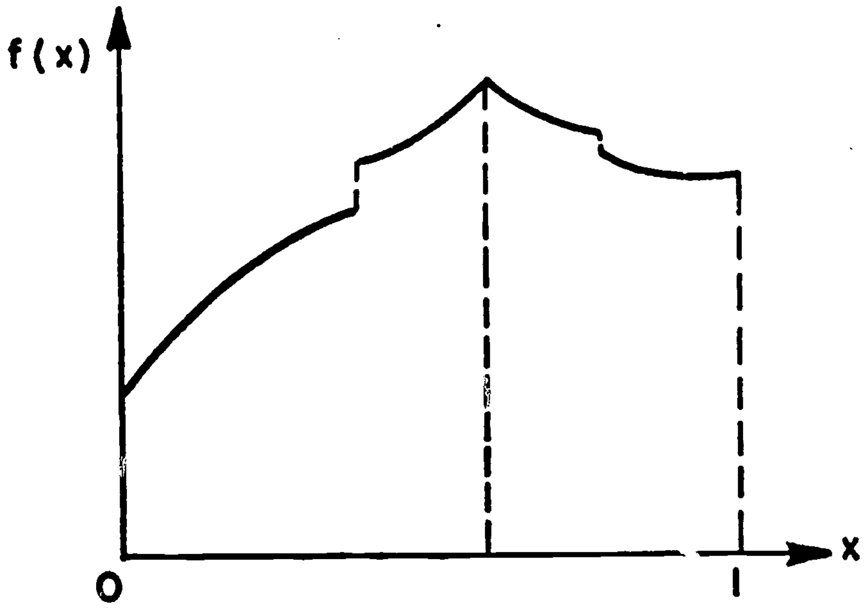


Figure 2: A Function Unimodal on the Interval [0,1].

These sequential methods all involve the computation of the function at one or more arguments which, because of the unimodality assumption, permit reduction of the length of the interval of uncertainty. A second set of functional values is then established for the reduced interval and yet a smaller uncertainty interval found. The process is repeated until the interval length is reduced to the specified value a. All these schemes have the property that the *number* of function evaluations required to achieve a specified terminal interval of uncertainty is known a priori, although the particular arguments used cannot be.

Clearly, the evaluation of the function at just one point on the interval yields insufficient information to reduce the interval of uncertainty. At least two function evaluations at different arguments in the interval must be made. The location of these arguments is arbitrary, but some choices are better than others. A few of these one-dimensional searching schemes are illustrated in Figures 3, 4 and 5.

If on the first pass of the sequential interval-slicing process we evaluate the function at two arguments centered about the midpoint and separated by a small distance ϵ, i. e., compute f(1/2+ϵ/2) and

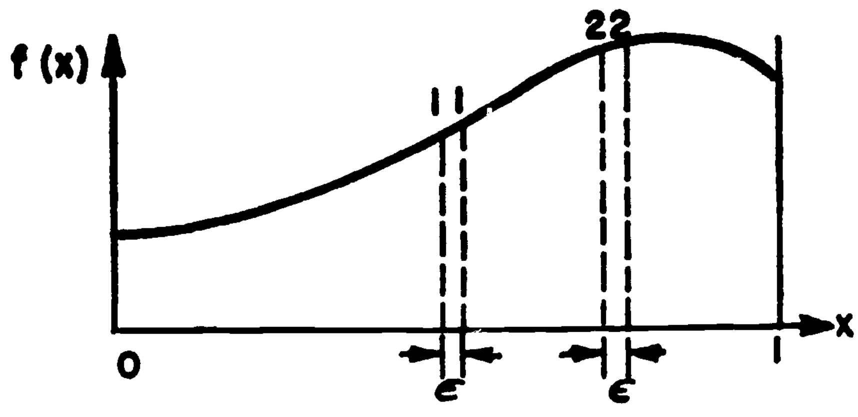


Figure 3: Dichotomous Search.

f(1/2 - ε/2 as shown in Figure 3, then the smaller sub-interval of length (1-ε)/2 can be rejected on the basis of the functional values (the 1's shown near the curve). In this case, the interval [(1-ε)/2,1] becomes the new interval of uncertainty; two new function evaluations are made at arguments centered about its midpoint (the 2's shown near the curve) and the interval subsequently reduced again based upon these new functional values. This process, which is termed the dichotomous search, reduces the interval of uncertainty approximately by half for each iteration of the procedure. N and α are related to the number of iterations n as follows:

$$\alpha = \frac{1}{2^n} + (1 - \frac{1}{2^n})\ \varepsilon$$

$$N = 2n = 2\ \frac{\ln\ (\frac{1 - \varepsilon}{\alpha - \varepsilon})}{\ln\ 2}\ \doteq 2.891n\ (\frac{1-\varepsilon}{\alpha-\varepsilon})$$

Other choices of argument locations for the evaluation of the f(x) on each iteration are illustrated in Figures 4 and 5. In both cases, the selected points are equally spaced in the interval. Figure 4 shows the two-point equal interval method applied to some unimodal function. On the first pass, the function is evaluated at x=1/3 and x=2/3) the 1's near the curve). That two-thirds of the original interval which must contain the extreme function value is retained as the new interval of uncertainty for the second pass; for the function shown, this would be the interval [1/3, 1]. On the next pass, the function is again evaluated at 1/3 and 2/3 of the span of the current interval, in this case at x = 5/9 and x = 7/9 (the 2's near the curve). Again, the 2/3 of the interval known to contain the extreme function value is saved, in this case the interval [5/9, 1].

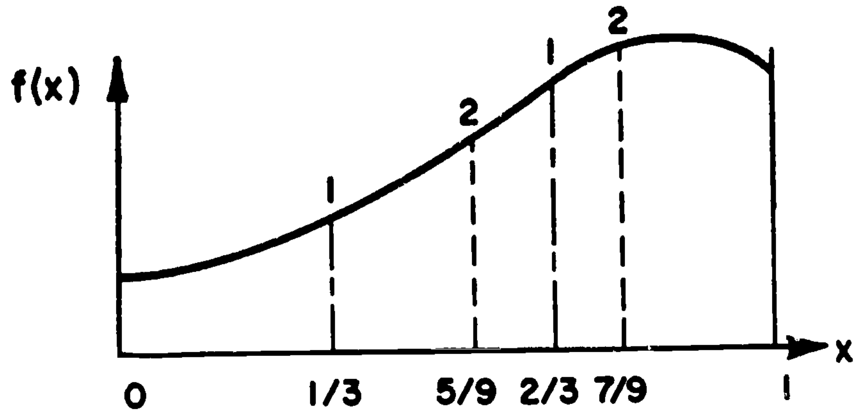


Figure 4: Two-Point Equal Interval Search.

The procedure is repeated until the interval of uncertainty is reduced to the specified α. The parameters α, n and N are related by:

$$\alpha = (\frac{2}{3})^n$$

$$N = - 4.95\ \ln\ (\alpha)$$

A similar three-point equal interval search is shown in Figure 5 below. In this case, the function is first evaluated at x=1/4, x=1/2 and $\bar{x}$=3/4 (the 1's near the curve). That half of the original interval centered about the argument of largest function value (x=3/4 in this case) is retained. In the second pass of the sequential scheme, the function is evaluated at x=5/8, x=3/4, and x=7/8. In this case, and in fact for all the odd-point equal interval methods, there will be one function value from the preceding iteration which will appear again and need not be recomputed [f(3/4) here]. The parameters α, n and N are related by:

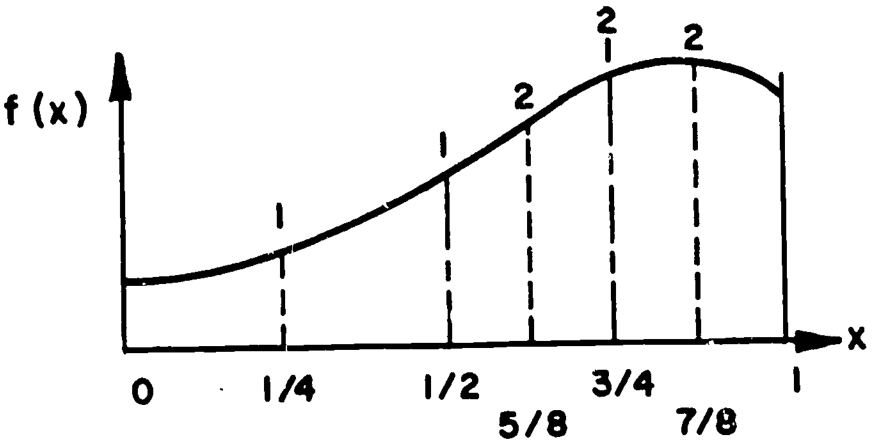


Figure 5: Three-Point Equal Interval Search.

$$\alpha = (\frac{1}{2})^n$$

$$N = 1 - 2.89\ \ln\ (\alpha)$$

It can be shown that the three-point equal interval search illustrated in Figure 5 is the most efficient (i. e., requires the fewest function evaluations N for a given α) of all possible equal interval search procedures.

Intuitively, one would guess that rejection of one-half of the uncertainty interval per iteration of a sequential interval-slicing procedure would be the best one could achieve on the average, given no advance information about f(x) except for its unimodal character. And from the viewpoint of the "minmax" strategy, i. e., the strategy which minimizes the likelihood of the worst possible outcome, this is the case. However, to reduce the uncertainty interval by half, using either the dichotomous or three-point equal interval search, requires *two* function evaluations. Are there other yet more efficient interval-slicing methods? Yes! Two such methods described below are called the golden section search and the Fibonacci search.

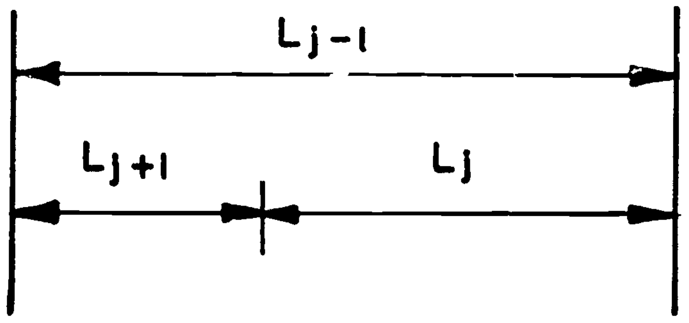
The golden section search is illustrated in Figure 6. In this strategy, two selected arguments are positioned symmetrically about the center of the interval of uncertainty in such a way that τ, the ratio of the lengths of successive intervals of uncertainty,

$$\tau = \frac{L_{j+1}}{L_j} = \frac{L_j}{L_{j-1}}$$

where $L_{j-1} = L_j + L_{j+1}$

remains constant and is approximately equal to 0.618, the so-called "golden-section" number. Hence, each iteration of this sequential technique reduces the interval by the factor 0.618, which would not appear to be as effective as the interval halving achieved by the methods described above. However, clc ıer examination shows that, after the first pass, one of the two arguments chosen for the j'th iteration will coincide with one of the arguments from the (j-1)'th iteration (e.g., x=0.618 in Figure 6). Thus, only one new function evaluation is required for all iterations after the initial one. The parameters a, n and N are related by:

$$a = (0.618)^n$$
$$N = n + 1 = 1 - 2.08 \ln (a).$$



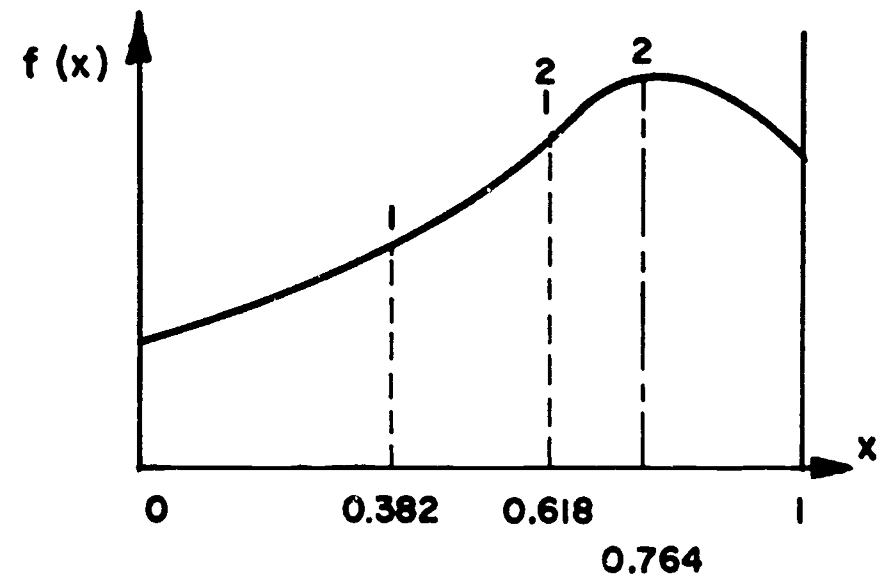


Figure 6: Golden Section Search.

It has been shown that the very best strategy for optimizing a unimodal function is the Fibonacci search, which is based on the Fibonacci number sequence F_i where

$$F_0 = 1$$

$$F_1 = 1$$

$$F_i = F_{i-1} + F_{i-2}, \quad i \geqslant 2 \; .$$

The first few numbers of the Fibonacci sequence generated by this recursion formula are:

i	F_i	i	F_i
0	1	5	8
1	1	6	13
2	2	7	21
3	3	8	34
4	5	9	55

The arguments at which the objective function f(x) is to be evaluated at each iteration of this sequential strategy are determined by ratios of appropriate Fibonacci numbers. Given a starting interval of length F_n, the Fibonacci search requires, at most, n function evaluations to reduce the interval of uncertainty to length 1.

To simplify the explanation of the method somewhat but without any real loss of generality, consider an objective function f(x) on the starting interval [0,13] shown in Figure 7. Suppose it is desired to reduce the length of the interval of uncertainty to 1. Then one examines the Fibonacci sequence, finds the F_n which corresponds to (or is next greater than) the length of the initial interval, 13. Since $F_6 = 13$, no more than n = 6 function evaluations will be required. The first two arguments are positioned by the ratios F_4/F_6 and F_5/F_6, namely, at 5/13 and 8/13 of the span of the interval, or at x = 5 and x = 8 (the 1's shown near the curve). Since f(5) > f(8), the new interval of uncertainty for the second pass of the iterative scheme is [0,8]. The arguments in the new interval are positioned by the ratios F_3/F_5 and F_4/F_5, i.e., at 3/8 and 5/8 of the span of the interval, or at x=3 and x = 5 (the 2's shown near the curve). In general, the ratios involved on the j'th iteration are

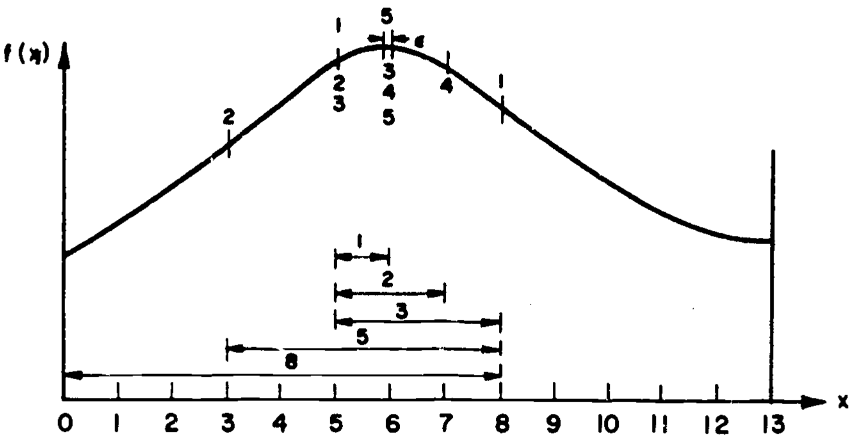


Figure 7: Fibonacci Search.

given by F_{n-j-1}/F_{n-j+1} and F_{n-j}/F_{n-j+1}. One of the arguments for the j'th pass will always coincide with one of the arguments for the (j-1)'th pass. Hence, just one new function evaluation need be made for each interation after the first. On the last (n=5)'th pass, one of the arguments is located within a small distance ε of the other to halve the interval of uncertainty after the (n-2=4)'th pass.

If the problem is reformulated in terms of the starting interval of length 1 used to illustrate the other searching strategies, the parameters α, n and N are related by:

$$\alpha = 1/F_n$$
$$N = n.$$

For $i > 5$, the ratio $F_{i-1}/F_i = 0.618$; thus, the method is closely related to the golden section search, and usually no more than one additional function evaluation can be saved by using the Fibonnaci search rather than the golden section search.

A closely related search procedure called the "lattice search" can be used to find the maximum of a discrete valued function which is unimodal.

A comparison of the number of function evaluations required to reduce an initial unit interval to a specified value of α is shown in Table III. The sequential strategies are clearly superior to the exhaustive search.

TABLE III

Comparison of Number of Function Evaluations

α	Exhaustive	3-Point Even Int.	Dichotomous*	Golden Section	Fibonacci
0.1	19	8	7	6	6
0.01	199	15	14	11	11
0.001	1999	21	20	16	16
0.0001	19999	28	27	21	20

*Depends on ε.

One might reasonably ask why so much emphasis has been placed on the one-dimensional unconstrained strategies, when most engineering design problems will have many design variables and probably constraints on the design variable values as well. The principal reason, which will become clearer later, is that many of the multidimensional strategies involve sequences of uni-directional searches, and these one-dimensional methods can be adapted very easily to handle the suboptimization problem. In addition, although these one-dimensional searching strategies have been developed without reference to constraints on the space of the variable, the selection of a starting interval places an arbitrary constraint on both the largest and smallest argument for which the objective function will be computed. Hence, the methods are directly applicable to constrained one-dimensional problems if the starting interval is chosen properly.

GEOMETRY OF MULTIDIMENSIONAL SURFACES

Some of the geometrical properties of a two-dimensional function $f(x_1,x_2) = f(\bar{x})$ are illustrated in Figures 8, 9 and 10. A point $\bar{x}$ in the two-dimensional space, i.e., in the x_1-x_2 plane of Figure 8, yields an observed value for the objective function $f(\bar{x})$ which is displayed in the third dimension, perpendicular to the space of the variables. The locus of objective function values in the third dimension is sometimes called the "response surface." The intersections of planes parallel to the x_1-x_2 plane (planes of constant function value) and the response surface, when projected onto the problem space of the x_1-x_2 plane, produce curves of constant function value called "level contours." Assuming that f(x) is everywhere continuous, the level contours will form connected curves; should the function be everywhere differentiable as well, then the level contours will be smooth curves (see Figure 8b).

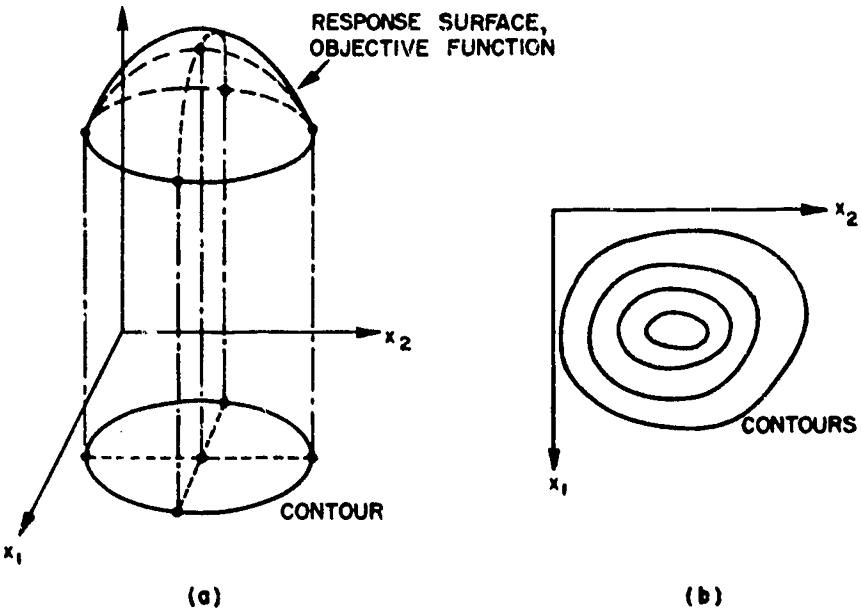


Figure 8: Response Surface and Level Contours for a Function of Two Variables.

If $f(\bar{x})$ is continuous and differentiable, it can be expanded about some point, say $\bar{a}$, (see Figure 9a) in a Taylor's series. If only the linear terms

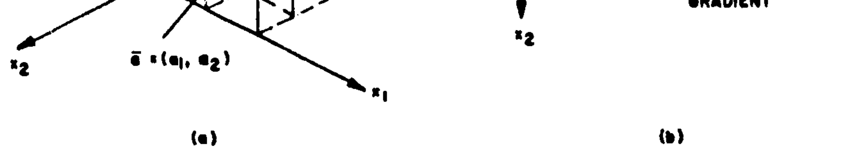

TANGENT PLANE AT $\bar{a}$
$y(\bar{x})$
TANGENT LINE IN THE $x_2 = a_2$ PLANE SLOPE $= f_{x_1}(\bar{a})$
TANGENT LINE IN THE $x_1 = a_1$ PLANE. SLOPE $= f_{x_2}(\bar{a})$
$f(\bar{a})$
x_2
$\bar{a} = (a_1, a_2)$
x_1
x_1
CONTOUR TANGENT
$\bar{a}$
$f(\bar{a})$
LEVEL CONTOUR
GRADIENT
x_2
(a)
(b)

Figure 9: Tangent Plane, Contour Tangent and Gradient for a Function of Two Variables.

in the series are retained, then the equation of the tangent plane of the response surface at $\bar{a}$ will be generated as

$$y(\bar{x}) = f(\bar{a}) + (x_1 - a_1)f_{x_1}(\bar{a}) + (x_2 - a_2)f_{x_2}(\bar{a})$$

If the intersection of the contour plane $f(\bar{a})$ and $y(x)$, the tangent plane at $\bar{a}$, is projected onto the x_1-x_2 plane, a line tangent to the level contour at $\bar{a}$ called the "contour tangent" is generated. Its equation is

$$(x_1 - a_1)f_{x_1}(\bar{a}) + (x_2 - a_2)f_{x_2}(\bar{a}) = 0 .$$

The projection onto the x_1-x_2 plane of that line in the tangent plane along which the function $y(\bar{x})$ increases most rapidly per unit distance, is called the "direction of steepest ascent" or simply the "gradient." The gradient at $\bar{a}$ is orthogonal to the contour tangent at $\bar{a}$ and its direction cosines d_1 and d_2 are given by:

$$d_1 = f_{x_1}(\bar{a})/S , \quad d_2 = f_{x_2}(\bar{a})/S$$

where $S = \sqrt{(f_{x_1}(\bar{a}))^2 + (f_{x_2}(\bar{a}))^2}$

The n-dimensional generalizations of the tangent plane and contour tangent are "hyperplanes," the n-dimensional analogue of a plane in three space. While impossible to picture graphically, their mathematical formulations follow in a simple way as:

$$y(\bar{x}) = f(\bar{a}) + \sum_{i=1}^{n} (x_i - a_i)f_{x_i}(\bar{a})$$

and

$$\sum_{i=1}^{n} (x_i - a_i)f_{x_i}(\bar{a}) = 0$$

The direction cosines of the gradient, which remains a line even in n-dimensional space, are given by

$$d_i = f_{x_i}(\bar{a})/S ,$$

where

$$S = \sqrt{\sum_{i=1}^{n} (f_{x_i}(\bar{a}))^2} , \quad i = 1,2,\ldots,n$$

A point $\bar{a}$ is a "stationary point" when all the first partial derivatives vanish, i.e , when $f_{x_i}(\bar{a}) = 0$, $i = 1,2,\ldots,n$. An "extreme point" is a stationary point at which the quadratic form

$$(\bar{x} - \bar{a})^T A(\bar{x} - \bar{a}) ,$$

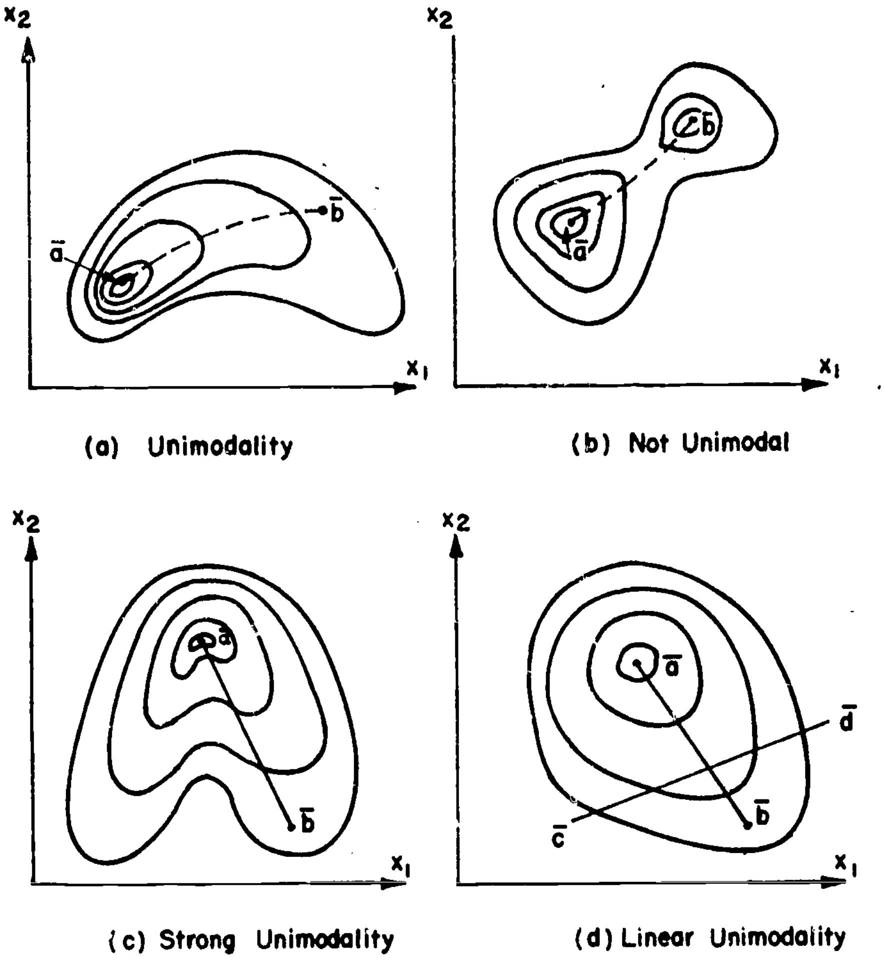
is definite. Here A is the matrix of second partial derivatives

$$A = \begin{bmatrix} f_{x_1x_2} & f_{x_1x_2} & \cdots & f_{x_1x_n} \\ f_{x_2x_1} & f_{x_2x_2} & & \\ \vdots & & & \\ f_{x_nx_1} & & \cdots & f_{x_nx_n} \end{bmatrix}_{\bar{a}}$$

If A is positive definite, the extreme point is a minimum point; if negative definite, the extreme point is a maximum point.

The concept of unimodality carries over from one dimension into n-dimensional space in a natural way; unimodality implies that the response surface has just one extreme point in the region under consideration. Three kinds of unimodality are illustrated in Figure 10. A function is said to be unimodal if, given an extreme point $\bar{a}$, there is *some* path from every other point $\bar{x}$ in the space along which the objective function values strictly rise or fall (see Figure 10a). Figure 10b shows the level contours for some two-dimensional function. Given an extreme point $\bar{a}$, it is not possible to find a strictly rising or falling path from point $\bar{b}$ to point $\bar{a}$; therefore the function is not unimodal. A somewhat more stringent form of unimodality called "strong unimodality" requires that the *straight line* path from every point $\bar{x}$ to the extreme point $\bar{a}$ be strictly rising or falling. This is illustrated in Figure 10c. A yet more restrictive form of unimodality, "linear unimodality," is illustrated in Figure 10d. In this case, all straight line paths between any two points in the space (e. g., $\overline{ab}$, $\overline{cd}$) must yield objective function values which are unimodal.

While a strictly unimodal function would have just one "local" extreme point equivalent to the one and only "global" extreme point, it is often possible to confine observation to a small region of interest in which the only concern is that the local behavior of the function be unimodal. Most of the optimization procedures to be described in the next section require that the function exhibit unimodal character, at least locally; some require more restrictive geometric properties. Virtually all are more efficient when the function is strongly or linearly unimodal.



(a) Unimodality (b) Not Unimodal

(c) Strong Unimodality (d) Linear Unimodality

Figure 10: Unimodality for a Function of Two Variables.

OPTIMIZATION METHODS FOR UNCONSTRAINED FUNCTIONS OF n VARIABLES ($n > 1$)

Several optimization methods suitable for finding extreme values for objective functions of more than one variable are shown in Table IV.

The calculus is the classical tool for solving extreme problems in n dimensions. Unfortunately, its usefulness in the engineering design area is rather limited because of the difficulty of evaluating high order derivatives and subsequently of solving systems of usually quite nonlinear simultaneous equations.

TABLE IV

Optimization Methods for Unconstrained Functions of n Variables ($n>1$) - $f(\bar{x})$

I. The Calculus

A. Root-Finding Procedures for f_{x_i} $(\bar{x})$

1. Newton-Raphson Method

II. Numerical Mathematics

III. Searching Schemes

A. Simultaneous Methods

1. Exhaustive Search
2. Random Search

B. Sequential Methods

1. Methods Requiring Evaluation of $f(\bar{x})$ only

a. Imbedded Random Search
b. Lattice Search
c. Univariate Search
d. Rotating Coordinate Method
e. Direct Search

2. Methods Requiring Evaluation of f_{x_i} $(\bar{x})$

a. Contour Tangent Method
b. Steepest Descent Methods
c. Optimal Steepest Descent Methods
d. Partan Methods

Since one of the criteria for finding a stationary point $\bar{a}$ of a continuous and differential function is that all the first partial derivatives vanish there, it may be possible to use one of the numerical root-finding procedures to solve the generated system of nonlinear equations fx_i $(\bar{a}) = 0,1 = 1,2,..,n$ for the unknown $a_1,a_2,...,a_n$. The Newton-Raphson procedure is a common iterative solution method for systems of nonlinear simultaneous equations; it has excellent convergence properties (when it converges) and is probably the best method when applicable. Unfortunately, the method requires that all second partial derivatives fx_ix_j $(\bar{x})$ be computable, preferably from analytical expressions. Numerical approximations of the required derivatives may be adequate in some cases.

Fortunately, all of the simultaneous searching methods and some of the sequential searching methods listed in Table IV only require evaluation of $f(\bar{x})$. Hence, these methods are particularly useful where derivatives are difficult to evaluate, the usual case in engineering design work. The multidimensional methods will be discussed in the order shown in Table IV.

The n-dimensional analogue of the one-dimensional exhaustive search already discussed requires the evaluation of the objective function at all nodes of a grid structure overlaid in the region of interest. For an n-dimensional hypercube (the n-dimensional equivalent of a rectangular area in two space or of a cube in three space) where α_i is the desired interval of uncertainty for variable x_i and $a_i \leq x_i \leq b_i$, the total number of function evaluations required is

$$N = \prod_{i=1}^{n} \left[\frac{2(b_i - a_i)}{\alpha_i} - 1 \right]$$

Note that even for a three-dimensional problem with a rather modest starting hypercube (say $b_i - a_i = 1$) and rather large interval of uncertainty (say $\alpha_i = 0.001$), the total number of function evaluations required is about 8×10^9; from a practical standpoint this approach is virtually useless.

For the general n-dimensional problem, assume that the feasible region is an n-dimensional hypercube with sides of length d_i in variable x_i, and that the final desired interval of uncertainty in variable x_i is α_i. Then the ratio of the volume of the final "hypercube of uncertainty" to the starting hypercube is given by

$$a = \prod_{i=1}^{n} \frac{\alpha_i}{d_i}$$

If we now choose a random point $\bar{X}$, i.e., a point in the space for which the n coordinates have been selected randomly, then a is also the probability that $\bar{X}$ is the hypercube of uncertainty containing the extreme value of the function. The probability that a random $\bar{X}$ is not in this hypercube is 1 - a; and the probability that not one of p such randomly generated points will fall into the optimal hypercube is $(1-a)^p$. Then the probability s that at least one of the p random $\bar{X}$ values will be in the optimal hypercube is

$$s = 1 - (1 - a)^p.$$

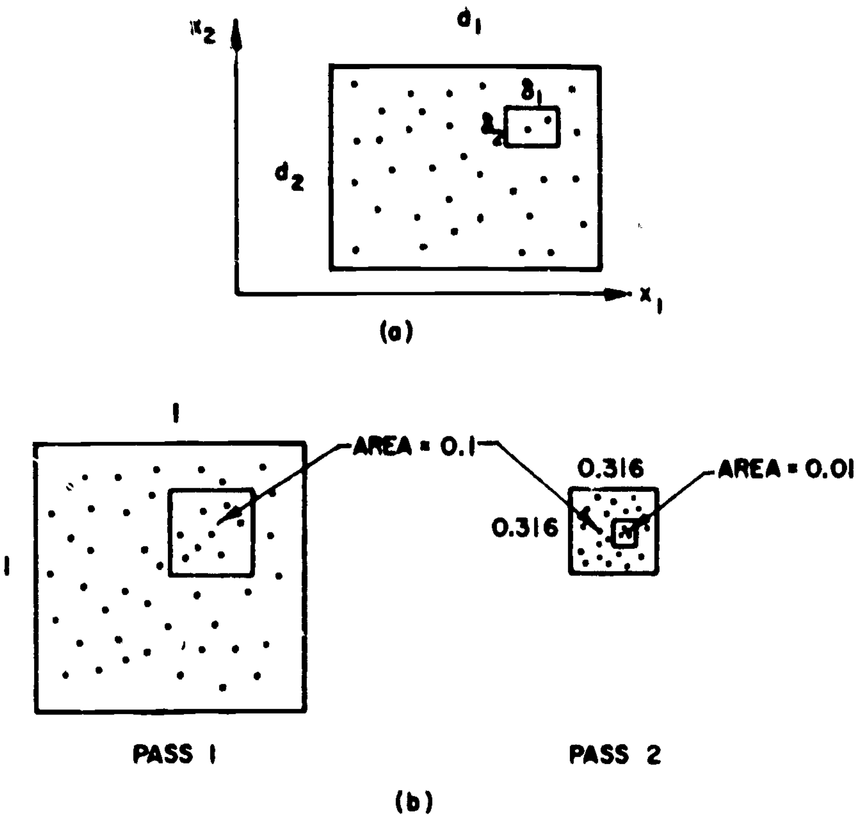


Figure 11: (a) Random Search. (b) Imbedded Random Search.

Assuming that the α_i are much smaller than the corresponding d^i,

$$p = -\frac{\ln(1 - s)}{a}$$

Some values of p for selected values of a and s are shown in Table V. An illustration of this "random search" approach for a two-dimensional problem is illustrated in Figure 11a.

TABLE V

p (No. of Random Points)

a	s=0.8	s=0.9	s=0.95	s=0.99
0.1	16	22	29	44
0.05	32	45	59	90
0.025	64	91	119	182
0.010	161	230	299	459
0.005	322	460	598	919

As described above, the number of function evaluations required for the random search increases exponentially with the number of dimensions, hence it is no real improvement over the exhaustive strategy described earlier.

A great reduction in the number of random points required with only a marginal decrease in the overall probability of success can be effected by imbedding the random search to give it the sequential character illustrated in Figure 11b. Here the original rectangle of uncertainty with sides of length 1 and area 1 may be reduced to a smaller rectangle of uncertainty with sides of length 0.316 and area 0.1, using just 44 function evaluations with a probability of success 0.99 (see Table V). If this subregion centered about the best point (based on the function value there) in the first pass is subsequently isolated and an additional 44 random points are generated in it, then a second rectangle of uncertainty with sides of length 0.1 and area 0.01 can be generated with a probability of success of 0.99. The overall probability for success will be degraded somewhat to $(0.99)^2 \doteq 0.98$, but the total number of function evaluations required is only 88. To achieve a comparable likelihood of success using just one iteration of the procedure would require 392 function evaluations.

The imbedding process can be carried out indefinitely with some decrease in the overall s at each iteration. There are no restrictions on the character of the objective function, i. e., differentiability, continuity and unimodality are not required. However, since the only criterion for choosing a given random point as the center of a new block for further imbedding is the value of the objective function there, the probability holds strictly only when the function value at every point in the hypercube of uncertainty about the extreme point is larger than at every point outside this hypercube.

SEQUENTIAL METHODS REQUIRING EVALUATION OF f(x) ONLY

In the following section, the lattice, univariate, rotating coordinate and direct search methods, which do not require computation of derivatives of the objective function, will be described briefly.

Lattice Method: In this method a grid of selected coarseness is overlaid in the region of interest (see Figure 12). A node of the grid is selected as a starting point. The function is evaluated there and at the 3^n-1 "adjacent" node points where n is the number of dimensions. The point with the greatest (or least) functional value is then selected as the next starting point and the procedure repeated, except where the functional values already computed need not be recomputed.

The process is continued until the greatest (or least) functional value occurs at the central point (starting point). In this case, the grid size is halved in each dimension and the search begun with the reduced grid mesh. The grid size reduction is repeated, when necessary, until the inter-point distance in each dimension is smaller than some specified tolerance value.

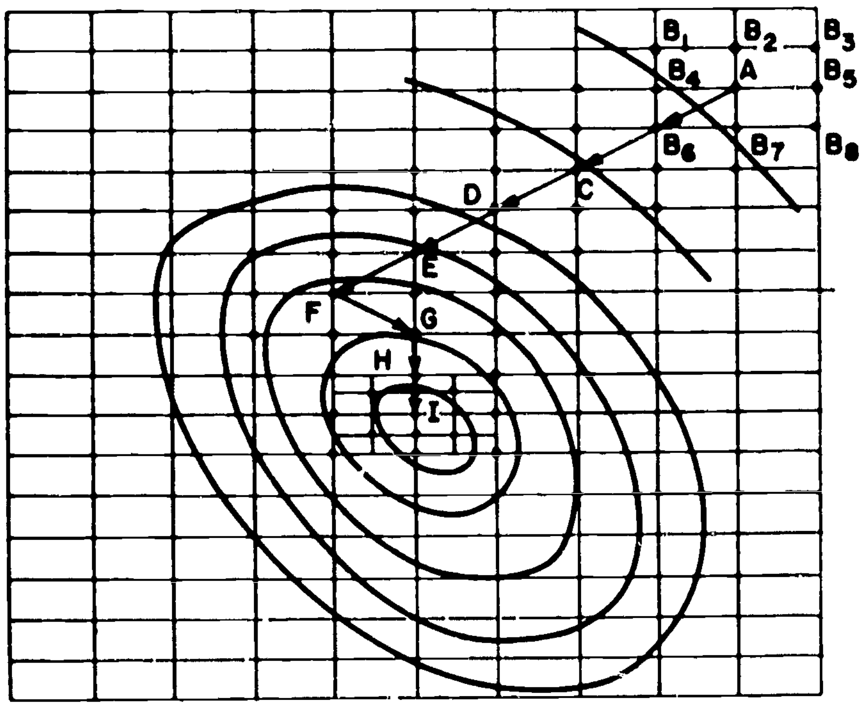


Figure 12: Lattice Search.

Univariate Search: In this search only one variable is changed at a time. Typically the variables are changed in order, i.e., $x_1, x_2, \dots, x^n$, so that the function is minimized (or maximized) in the coordinate directions. Starting with some arbitrary point $\bar{x}_o$, successive intermediate points $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ are found by carrying out linear minimizations in the directions

$$\bar{p}_o = \begin{bmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix}, \bar{p}_1 = \begin{bmatrix} 0 \\ 1 \\ \cdot \\ \cdot \\ 0 \end{bmatrix}, \dots \bar{p}_{n-1} = \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ 0 \\ 1 \end{bmatrix}$$

such that $\bar{x}_{i+1}$ is the point of minimum functional value along the one in direction $\bar{p}_i$ passing through point $\bar{x}_i$. Thus

$$\bar{x}_{i=1} = \bar{x}_i + a_i \bar{p}_i, \quad i = 0,1,\dots,n-1$$

where $|a_i|$ is the distance between $\bar{x}_i$ and $\bar{x}_{i+1}$.

Although one need not compute the gradient

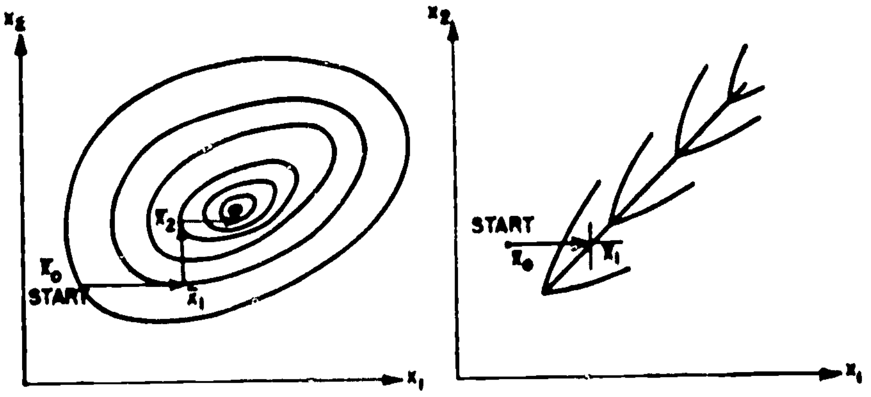
$$\bar{g}(\bar{x}_{i+1}) = \bar{g}_{i+1}$$

to find a_i, a_i is determined by the relationship

$$\bar{p}_i^T \bar{g}_{i+1} = 0$$

i.e., $\bar{p}_i$ is tangent to the contour at $\bar{x}_{i+1}$. After completing one round, a second round is completed, etc., until all $|a_i|$ are smaller than some corresponding tolerance values ϵ_i.

The method can be defeated if the variables $x_1, \dots, x_n$ interact strongly (Figure 13,) such as with a ridge-like structure for which the normalized gradient vector on the ridge has all direction cosines approximately equal in magnitude. If there is no interaction at all, i. e., if the principal axes of the surface are in the coordinate directions, then the method will find the extreme value in the least number of rounds.



(a) Little interaction. (b) Ridge with strong interaction.

Figure 13: Univariate Search.

Rotating coordinate method: In an attempt to overcome the ridge orientation problem of the univariate search procedure, Rosenbrock[2] developed a variant of the univariate method which involves rotation of the axes of the search.

[2] H.H.Rosenbrock, "An Automatic Method for Finding the Least Value of a Function," The Computer Journal, Vol. 3, p. 175 (1960).

Here the procedure begins with a standard one-round univariate search starting at point $\bar{x}_0$ (arbitrary), i.e., n successive linear minimizations are carried out in the coordinate directions

$$\bar{x}_{i+1} = \bar{x}_i + \alpha_i \bar{p}_i, \quad i = 0, 1, \ldots, n-1$$

such that

$$\bar{p}_i^T \bar{g}_{i+1} = 0$$

where

$$\bar{p}_i = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ 1 \\ \cdot \\ \cdot \\ 0 \end{bmatrix} \leftarrow \text{i'th position}$$

Next, a new set of searching directions is generated, such that

$$\bar{p}_n = \frac{\bar{x}_n - \bar{x}_o}{||\bar{x}_n - \bar{x}_o||}$$

and $\bar{p}_{n+1}, \ldots, \bar{p}_{2n-1}$

satisfy the orthogonality conditions

$$\bar{p}_i^T \bar{p}_j = 0, \quad i,j = n, n+1, \ldots, 2n-1 \quad i \neq j$$

and

$$\bar{p}_i^{-T} \bar{g}_{i+1} = 0.$$

As before,

$$\bar{x}_{i+1} = \bar{x}_i + \alpha_i \bar{p}_i$$

In general, each round of the method consists of n one-dimensional searches in n orthogonal directions. In the first round the n directions used are the coordinate directions. In the second round the first search is conducted in the direction given by a line passing through points $\bar{x}_o$ and$\bar{x}_n$; the following n-1 searches are in directions mutually orthogonal and orthogonal to the first one as well. In successive rounds new sets of orthogonal vectors $\bar{p}_n, \ldots, \bar{p}_{2n-1}$ are generated so that for round k + 1

$$\bar{p}_{nk} = \frac{\bar{x}_{nk} - \bar{x}_{n(k-1)}}{||\bar{x}_{nk} - \bar{x}_{n(k-1)}||}, \quad k = 1, 2, \ldots.$$

The additional n-1 search direction vectors are then generated using the Gram Schmitt method.

The coordinate rotation technique used here causes the vectors $\bar{p}_{nk}$ to align themselves along the direction of a sharp ridge, and hence the method tends to be quite good for tracking along a ridge, particularly along a relatively straight one. The method can be defeated however, given a bad starting point $\bar{x}_o$.

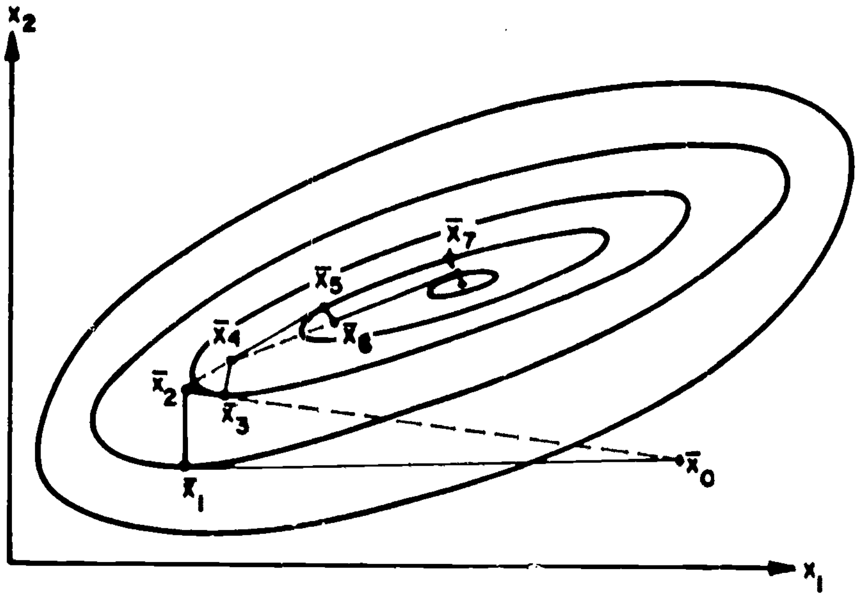


Figure 14: Rotating Coordinate Method.

Direct search: The direct search of Hooke and Jeeves[3] is a "trial-and-error" technique which has been used successfully by many investigators. There appears to be no one set of rules which constitute "direct search." All the approaches consist of a local "pattern search" followed by a global move called a "pattern move."

The exploratory search consists of a restricted univariate search, such that at most one step (usually small) of length δ_j is taken in the j'th direction.

1. Start with an arbitrary base point $\bar{b}_1$.
2. Add (or subtract) δ_1 to x_1 at $\bar{b}_1$ and evaluate the function there. Let

$$\bar{\delta}_1 = \begin{bmatrix} \delta_1 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

and evaluate the function at $\bar{b}_1 + \bar{\delta}_1$ if necessary at $\bar{b}_1 - \bar{\delta}_1$ and at $\bar{b}_1$ keeping the "best" point. Let the new "temporary" point be called $\bar{t}_{11}$, i.e.,

[3] R. Hooke and T. A. Jeeves, "Direct Search Solution of Numerical and Statistical Problems," J. A. C. M., 8, 2, 212-229 (1961).

$$\bar{t}_{11} = \begin{cases} \bar{b}_1 + \bar{\delta}_1 & \text{if } f(\bar{b}_1+\bar{\delta}_1) < f(\bar{b}_1) \\ \bar{b}_1 - \bar{\delta}_1 & \text{if } f(\bar{b}_1-\bar{\delta}_1) < f(\bar{b}_1) \\ \bar{b}_1 & \text{if } f(\bar{b}_1) < \min\{f(\bar{b}_1+\bar{\delta}_1), f(\bar{b}_1-\bar{\delta}_1)\} \end{cases}$$

3. Next add (or subtract) δ_2 to x_2 at $\bar{t}_{11}$ where

$$\bar{\delta}_2 = \begin{bmatrix} 0 \\ \delta_2 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

retaining the best point. Call it $\bar{t}_{12}$, i.e.

$$\bar{t}_{12} = \begin{cases} \bar{t}_{11} + \bar{\delta}_2 & \text{if } f(\bar{t}_{11} + \bar{\delta}_2) < f(\bar{t}_{11}) \\ \bar{t}_{11} - \bar{\delta}_1 & \text{if } f(\bar{t}_{11} - \bar{\delta}_2) < f(\bar{t}_{11}) \\ \bar{t}_{11} & \text{if } f(\bar{t}_{11}) < \min\ \{f(\bar{t}_{11} + \bar{\delta}_2), f(\bar{t}_{11}-\bar{\delta}_2)\} \end{cases}$$

4. Continue this process for each of the n variables $x_1, x_2, \ldots x_n$, each time finding a new temporary point $\bar{t}_{1j}$ where

$$\bar{t}_{1j} = \begin{cases} \bar{t}_{1,j-1}+\bar{\delta}_j & \text{if } f(\bar{t}_{1,j-1}+\bar{\delta}_j) < f(\bar{t}_{1,j-1}) \\ \bar{t}_{1,j-1}-\bar{\delta}_j & \text{if } f(\bar{t}_{1,j-1}-\bar{\delta}_j) < f(\bar{t}_{1,j-1}) \\ \bar{t}_{1,j-1} & \text{if } f(\bar{t}_{1,j-1}) < \min\{(\bar{t}_{1,j-1}+\bar{\delta}_j)f(\bar{t}_{1,j-1}+ \bar{\delta}_j)\} \end{cases}$$

$$\bar{\delta}_j = \begin{vmatrix} 0 \\ 0 \\ \delta_j \\ 0 \\ 0 \end{vmatrix} \quad \leftarrow \text{j'th element}$$

Let the second base point $\bar{b}_2$ be equal to $\bar{t}_{1,n}$. Note that the first exploratory search around the starting point $\bar{b}_1$ requires at least n + 1 but no more than 2n+1 function evaluations.

5. Next, make the first pattern move by moving in the direction $\bar{b}_2-\bar{b}_1$ along the line from $\bar{b}_1$ to $\bar{b}_2$ a distance equal to <u>twice</u> the distance from $\bar{b}_1$ to $\bar{b}_2$. Call this point temporary point $\bar{t}_{20}$, i.e.

$$\bar{t}_{20} = \bar{b}_1 + 2(\bar{b}_2-\bar{b}_1) = 2\bar{b}_2 - \bar{b}_1.$$

6. Now conduct an exploratory pattern search about the point $\bar{t}_{20}$ as was done around the original base point $\bar{b}_1$ (see 2, 3, and 4 above.) Call the terminal point of the search $\bar{t}_{2n}$ and let

$$\bar{b}_3 = \bar{t}_{2n}$$

7. Make the second pattern move in the direction $\bar{b}_3 - \bar{b}_2$ to locate a third temporary point $\bar{t}_{30}$

$$\bar{t}_{30} = \bar{b}_2 + 2(\bar{b}_3-\bar{b}_2) = 2\bar{b}_3-\bar{b}_2$$

8. This process of conducting exploratory searches around the points $\bar{t}_{ko}$ and determining the k'th base point as $\bar{b}_{k+1} = \bar{t}_{k_n}$ followed by a pattern move to the point $\bar{t}_{k+1,0}$ according to $\bar{t}_{k+1,0} = 2\bar{b}_{k+1} - \bar{b}_k$ continues so long as the function value at $\bar{t}_{k+1}$ is smaller than at $\bar{b}_{k+1}$.

9. Should $f(\bar{t}_{k+1}) > f(\bar{b}_{k+1})$, then the base point $\bar{b}_k$ is dropped from consideration, and a new search is started by designating

$$\bar{b}^*_1 = \bar{b}_{k+1}$$

and returning to step 2.

10. Should a local exploratory search near the newly designated $\bar{b}^*_1$ fail to find a temporary point $\bar{t}^*_{1n}$ different from $\bar{b}^*_1$, i.e., should $\bar{b}^*_2 = \bar{b}^*_1$ then the sizes of the search increments δ_j must be reduced and a new exploratory search made by returning to step 2.

11. When the δ_j have been reduced sufficiently to be "small enough," i.e., when

$$\delta_j < \varepsilon_j$$

where ε_j is some small number, then the search is completed and the last designated $\bar{b}_1$ becomes the direct search approximation to the extreme point.

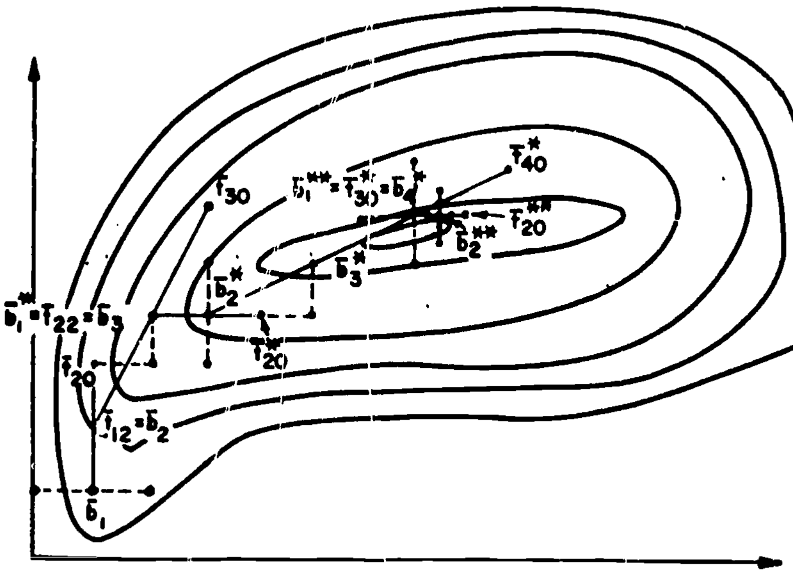


Figure 15: Direct Search Method.

The method appears to be very efficient for many minimization problems, particularly those where ridges are encountered. The direction of successive pattern moves tends to become aligned with the ridge. Experimentally in such problems, the number of function evaluations required tends to increase directly with n (the number of dimensions) rather than with some higher power of n as would be expected.

A variant of the procedure described above computes the temporary points as follows:

$$\bar{t}_{k+1,0} = 2\bar{b}_{k+1} - \bar{b}_1 \ .$$

Thus all pattern moves are along lines with directions $\bar{b}_{k+1}$-$\bar{b}_1$ which pass through $\bar{b}_1$. The step sizes in the pattern moves are thus amplified more rapidly than in the standard method.

It should be noted that since the criterion for termination is failure of a univariate pattern search using sufficiently small δ_j, the method can fail in those cases where a univariate search would fail, i. e., for surfaces with sharp ridges having certain orientations with respect to the coordinate axes.

SEQUENTIAL METHODS REQUIRING THE EVALUATION OF FIRST PARTIAL DERIVATIVES

In the following section, methods requiring the computation of first partial derivatives or the gradient of the objective function f(x) will be discussed. Where possible, the derivatives should be computed from analytical expressions. Since this is usually not feasible when dealing with complicated objective functions (e. g., the cost of a plant), one must usually compute the derivatives numerically by perturbing each variable in turn and computing the value of the objective function there. To compute all n first partial derivatives numerically (n being the dimension of the space) will usually require n + 1 function evaluations.

Method of Contour Tangents: If the function to be minimized (or maximized) is strongly unimodal, then the contour tangent line in two space (a plane or hyperplane in higher dimensional space) separates the space into two half-spaces (Figure 16). The functional extreme point will lie in one of these two half-spaces. One "blocking" approach, called the contour tangent method, begins with the arbitrary selection of a starting point $\bar{x}_0$, computation of the first partial derivatives there, and subsequent determination of the contour tangent. Using the direction of the gradient as the criterion, one half of the total space, the half on the "wrong" side of the contour tangent, can be rejected as not possibly containing the extreme point. A new point $\bar{x}_1$ located in the "center" of the saved half-space is selected next and the process repeated. With each pass of the method part of the space is rejected. When the bounded or "blocked" space known to contain the extreme point is small enough, the search is complete. At each point selected, $\bar{x}_i$, it is necessary to compute the equation of the contour tangent:

$$(\bar{x} - \bar{x}_i)^T \bar{g}(\bar{x}_i) = f(\bar{x}_i)$$

There are two major difficulties encountered in applying the method, selecting the locations of the $\bar{x}_i$, and maintaining a running record of the extent of the space known to contain the extreme point. Wilde[4] suggests four different points, the "midpoint," "minimax point," "median," and "centroid" for $\bar{x}_{i+1}$. From a practical standpoint only the midpoint can be calculated easily. Even so, the number of constraining hyperplanes may become very large, leading to an extremely complicated algorithm even for the two-dimensional case.

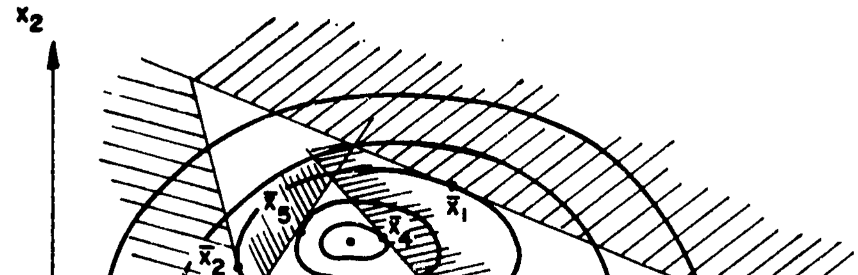


Figure 16: Method of Contour Tangents.

[4] D. J. Wilde, "Optimization by the Method of Contour Tangents," AICHE Journal 9, 2. p. 186 (1963).

Method of Steepest Descent: The successive points of the search $\bar{x}_{i+1}$ are in the direction of the negative gradient at

$$\bar{x}_i, -\bar{g}_i = -\bar{g}(\bar{x}_i)$$

$$\text{i.e., } \bar{x}_{i+1} = \bar{x}_i + \alpha_i \bar{p}_i, \; i=0, 1, \ldots$$

$$\text{where } \bar{p}_i = \frac{-\bar{g}_i}{\|\bar{g}_i\|} \text{ and } \alpha_i \text{ is arbitrary}$$

$$\text{but such that } f(\bar{x}_{i+1}) < f(\bar{x}_i)$$

(see Figure 17). Justification for moving in the negative gradient direction is that the greatest improvement in $f(\bar{x})$ per unit length of travel, at least for infinitesimal α_i, is in the negative gradient direction.

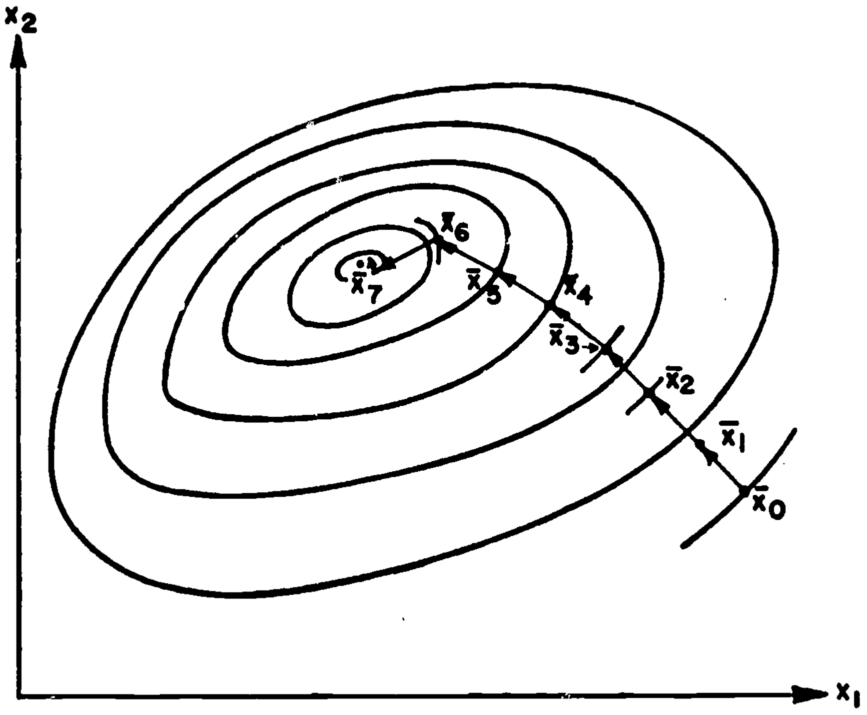


Figure 17: Method of Steepest Descent.

There are three principal difficulties with the method:

1. Criteria for choosing the α_i are not clear.
2. Scaling can cause problems.
3. It may take a very large number of steps to find the optimum value.

One suggestion made by Marquardt[5] which allows a more-or-less automatic step-size adjustment is to use the formula

$$\alpha_i = \alpha_{i-1}(a_1 + a_2 \cos^4 \Theta)$$

where Θ is the angle between $\bar{p}_i$ and $\bar{p}_{i-1}$ and a_1 and a_2 are constants (suggested values are $a_1 = 0.5$, $a_2 = 1.5$). Note that this causes $\alpha_i > \alpha_{i-1}$ when the gradient direction is not changing much (when $\bar{x}_i$ and $\bar{x}_{i-1}$ are far from the optimum).

The scaling problem can be illustrated by the following simple function

$$f(x,y) = x^2 + 25y^2$$

which, with the transformation $u = 5y$, becomes

$$f(x,u) = x^2 + u^2.$$

In the x,y space the gradient direction is not nearly the optimal direction (the direction toward the minimum) while in the x,u space it is the optimal direction. It is wise to scale the variables to make the contours as nearly circular, spherical, or hyperspherical as possible. Since one can't very well know what the shape of the contours is ahead of time, such a scaling is generally not possible. One alternative is to choose a starting hypercube of uncertainty and then use the length of the i'th edge as a normalizing factor for the i'th variable. Scaling difficulties are almost always encountered when the variables are dimensionally incompatible (pressure and temperature, for example). All the gradient dependent methods which follow also have scaling problems.

Optimal Steepest Descent Method: This perhaps misnamed method is similar to the method of steepest decent, except that the α_i are determined automatically:

$$\bar{x}_{i+1} = \bar{x}_i + \alpha_i \bar{p}_i$$

$$\text{where } \bar{p}_i = -\frac{\bar{g}_i}{|\bar{g}_i|}$$

$$\bar{p}_i^T \bar{g}_{i+1} = 0$$

Thus the gradient direction at $\bar{x}_i$ is followed until the gradient of $f(\bar{x})$ is orthogonal to it (see Figure 18).

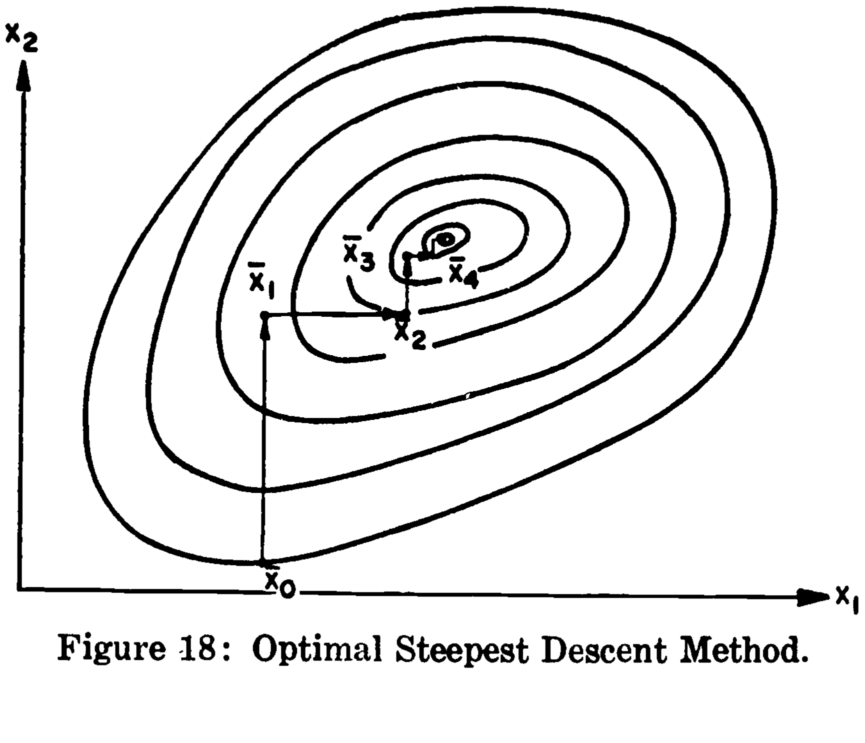


Figure 18: Optimal Steepest Descent Method.

[5] Marquardt, Chem. Eng. Progress, Vol. 55, No. 6 (June, 1959).

A linear minimization in the direction $\bar{p}_i$ must be carried out starting at point $\bar{x}_i$. Any of the one-dimensional searching strategies may be employed to find $\bar{x}_{i+1}$, the point of minimum functional value in the negative gradient direction. The method usually requires fewer functional evaluations than the method of steepest descent. This is particularly true for those problems which require that the partial derivatives be computed numerically.

PARTAN Methods: Because the steepest descent methods often exhibit slow convergence to the extreme point, oscillating severely from one step to the next, particularly when the objective function surface possesses deep, narrow valleys, so-called acceleration methods have been developed which usually speed up convergence greatly. The methods are motivated primarily by some geometric properties of quadratic functions. Since most continuous and differentiable functions assume a near-quadratic character in the immediate vicinity of an extreme point, the methods can often be used for non-quadratic functions as well.

Consider the simple two-dimensional quadratic function:

$$f(x_1, x_2) = a_{11}X_1^2 + a_{12} X_1 X_2 + a_{22} X_2^2$$

where $X_1 = x_1 - x_1^*$, $X_2 = x_2 - x_2^*$ and

$\bar{x}^* = \begin{vmatrix} x_1^* \\ x_2^* \end{vmatrix}$ is the center (extreme)

point of $f(x_1, x_2)$

It can be shown relatively easily that if $f(\bar{x})$ is quadratic, then the contour tangents at $\bar{x}_1$ and $\bar{x}_2$ are parallel. Conversely, it can be shown that, if one has two parallel lines CT_1 and CT_2 and locates the point of minimum function value on each (say $\bar{x}_1$ and $\bar{x}_2$), then the functional minimum is on the line passing through $\bar{x}_1$ and $\bar{x}_2$ (see Figure 19).

In two dimensions the PARTAN method usually takes on one of the following forms:

Method (a) (see Figure 19):

1. Arbitrarily choose a line CT_1 in the space of the independent variables.
2. Choose any other line CT_2 parallel to CT_1.
3. Find the "best" point on CT_1 using one of the one-dimensional search techniques. The best point (call it $\bar{x}_1$) is the point for which the line CT_1 is tangent to a contour of the objective function.
4. Repeat step 3 for line CT_2 and call the best point $\bar{x}_2$.
5. Search in the direction of the line passing through $\bar{x}_1$ and $\bar{x}_2$ using a one-dimensional search strategy. If the function is quadratic, the best point on this line will be the functional extreme point.

Method (b) (see Figure 20):

1. This method begins with two passes of the optimal gradient method as shown in the fig-

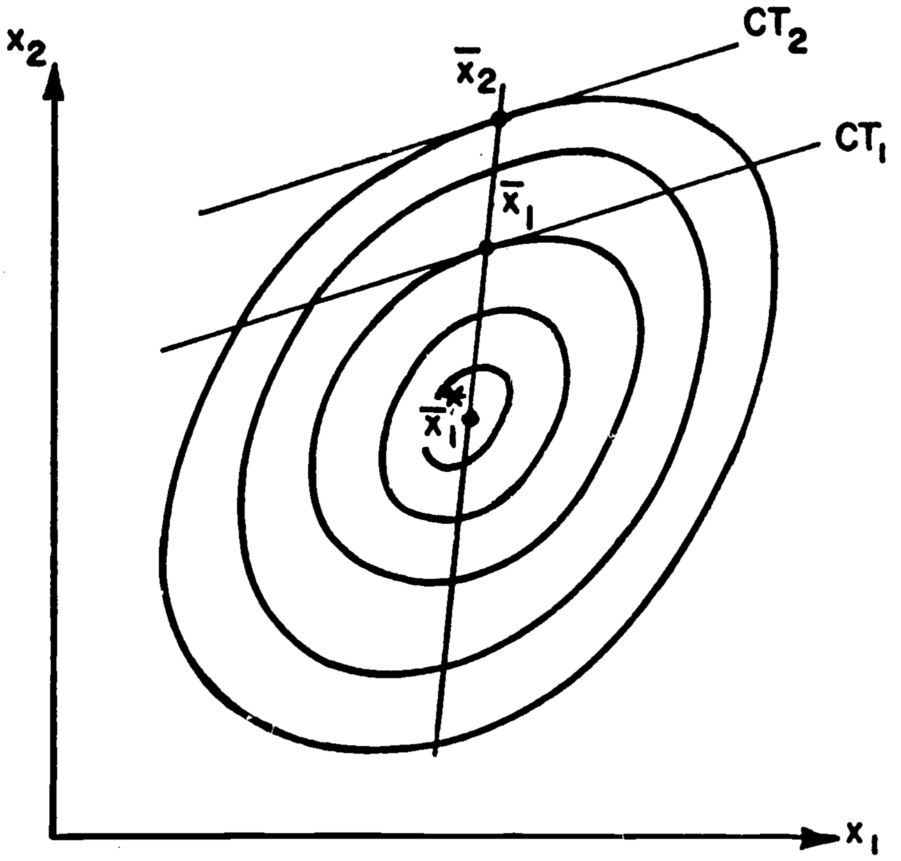


Figure 19: PARTAN Method. Method (a)

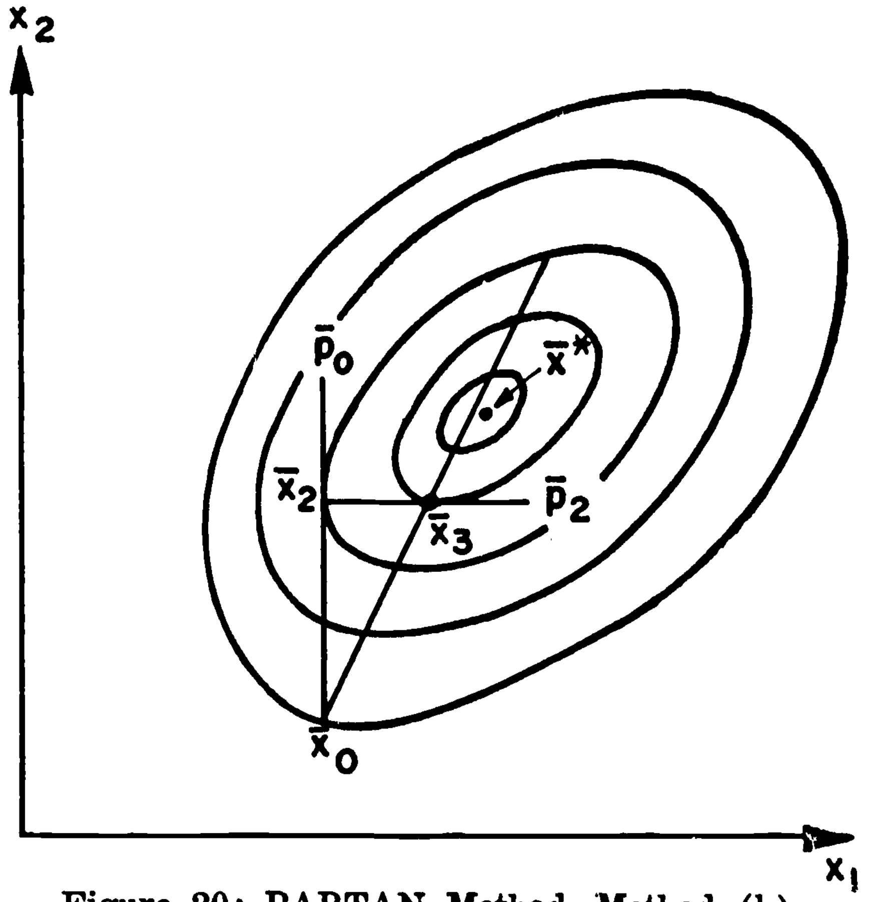


Figure 20: PARTAN Method. Method (b).

ure. Let $\bar{x}_o$ be the initial point and $\bar{x}_2$ and $\bar{x}_3$ be the terminal points for the first two optimal gradient searches.

$$\bar{x}_2 = \bar{x}_o + \alpha_o \bar{p}_o$$

$$\bar{p}_o = - \frac{\bar{g}_o}{\|\bar{g}_o\|}$$

$$\bar{p}_o^{\,T} \bar{g}_2 = 0$$

$$\bar{x}_3 = \bar{x}_2 + \alpha_2 \bar{p}_2$$

$$\bar{p}_2 = - \frac{\bar{g}_2}{\|\bar{g}_2\|}$$

$$\bar{p}_2^{\,T} \bar{g}_3 = 0$$

2. A final linear minimization is now performed in the direction $\bar{x}_o \bar{x}_3$ along the line passing through $\bar{x}_o$ and $\bar{x}_3$. The minimum point $\bar{x}^*$ will be the extreme point if $f(\bar{x})$ is a quadratic function. This follows because $\bar{p}_2$, the direction of the contour tangent at $\bar{x}_3$, is orthogonal to $\bar{p}_o$ and hence parallel to the contour tangent at $\bar{x}_o$.

OPTIMIZATION METHODS FOR CONSTRAINED OBJECTIVE FUNCTIONS

Figure 21 illustrates a typical optimization problem in two dimensions with one nonlinear constraint. Note that the unconstrained functional extreme point lies outside the region of feasible solutions.

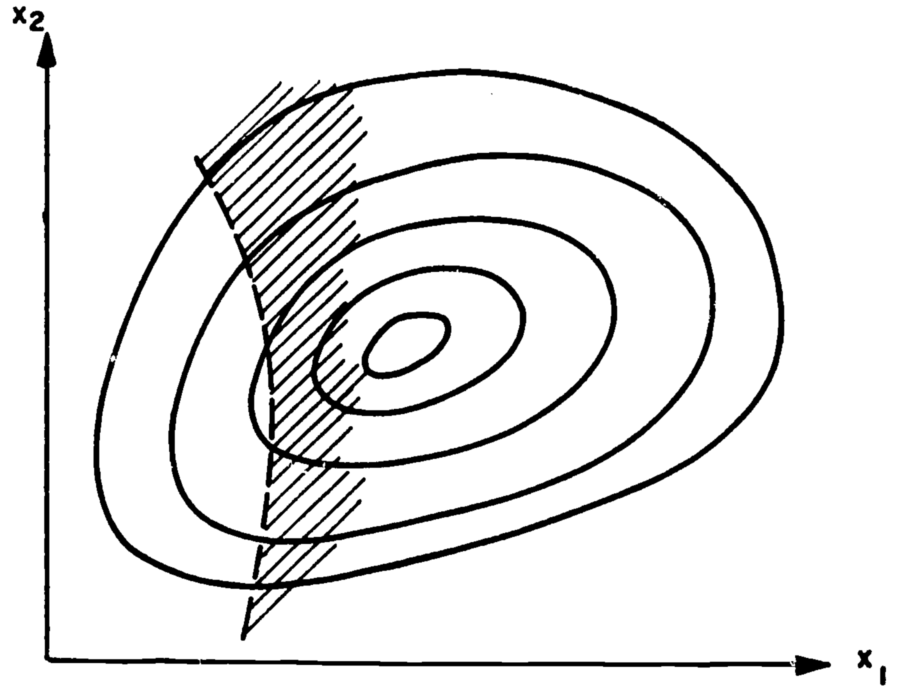


Figure 21: A Constrained Optimization Prcblem.

The methods available to the engineering designer for dealing with such problems vary in complexity and power, principally with the nature of the objective function and constraints. Table VI lists a few of the methods available.

TABLE VI

Optimization Methods for Constrained Objective Functions

I. Linear Programming
II. Nonlinear Programming
  A. Lagrange Multipliers
  B. Iterative Linear Programming (Simplicial Methods
    1. Cutting Plane Method
  C. Adaptations of Methods for Unconstrained Functions
    1. Gradient Projection Method
    2. Multiple Gradient Summation Method

Linear Programming

A linear programming problem is a constrained optimization problem for which the objective function and all constraints are linear in the problem variables. Very powerful algorithms have been developed for solution of such problems. Linear programming has in the recent past been the most used of all optimization methods, particularly in such business-oriented applications as resource allocation.

Many industrially important problems are amenable to solution by linear programming methods. Three of the more important types are known by the labels "The Transportation Problem," "The Activities Analysis Problem," and "The Diet Problem." A brief description of these three typical linear programming problems is given below.

a. **The Transportation Problem:** A company wishes to ship a number of units of some item from its warehouses to its customers (for example, a rubber company shipping tires to its wholesale distributors). Each customer has ordered a number of units of the item; each of the company's warehouses can supply up to a certain amount of its current inventory of that item. What shipping schedule should the company use to minimize the total transportation cost, assuming that the total inventory available from the warehouses is equal to the total of the orders of all customers?

b. **The Activities Analysis Problem:** A manufacturer has fixed amounts of several resources available for his use (these might be raw materials, labor, equipment, money, etc.). The resources can be combined in several different ways to make one or more of several different possible products. It takes a certain number of units of a particular resource to produce one unit of a particular product. He knows how much product he can make for every unit of any particular product he manufactures. How should he allocate available resources to maximize the total profit?

c. **The Diet Problem**: Given the nutrient contents of several foods and a minimum daily dietary requirement for each nutrient, minimize the total food cost while maintaining at least a minimum nutrition level.

Each of these problems can be written in the form of a standard linear-programming model of the form

$$\text{maximize} \quad \sum_{j=1}^{n} c_j x_j$$

subject to the constraints

$$x_j \geqslant 0 , \quad j = 1,2,\ldots,n$$

$$\sum_{j=1}^{n} a_{ij} x_j \begin{Bmatrix} \leq \\ = \\ \geq \end{Bmatrix} b_i, \; b_i \geq 0, \; i = 1,2,..m$$

The objective is to maximize the *linear* function of the n variables x_j, subject to m constraining relationships also *linear* in the x_j; the constraints can assume the form of equalities or inequalities ($\leqq$,$\geqq$). In addition, each x_j in the feasible solution must be non-negative.

In two dimensions the linear programming problem has a very simple geometric interpretation. Suppose the problem is to maximize the function $f(x_1,x_2) = 3x_1 + 4x_2$ subject to the linear constraints

$$\begin{aligned} 2x_1 + 5x_2 &\leqq 10 \\ 4x_1 + 3x_2 &\leqq 12 \\ x_1 &\geqq 0 \\ x_2 &\geqq 0 \end{aligned}$$

This is illustrated in Figure 22.

Notice that the inequalities require that the solution be in or on the boundaries of the cross-hatched region. This region is termed the "feasible" region and any (x_1, x_2) in the region is said to be a *feasible solution*. The linear inequalities enclose a feasible region which has the form of a *convex polygon* where "convex" implies that all points on the straight line joining any two points in the set of feasible points are also in the set. This simply means that a linear combination of two feasible solutions is also a feasible solution, provided the combination lies between the two solution points chosen.

For the given problem, the objective function is the family of straight lines with slope -3/4; its value decreases as the lines approach the origin. Note

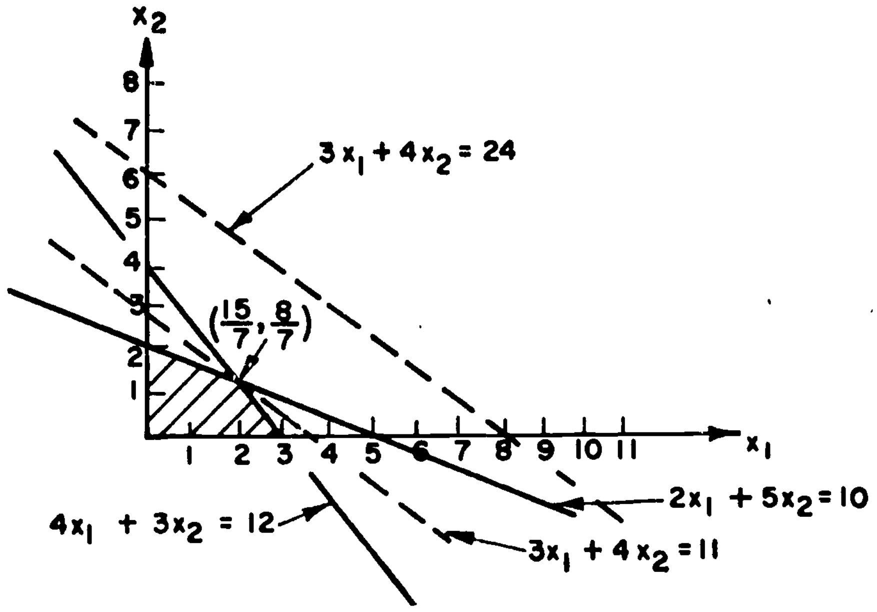


Figure 22: Geometric Interpretation of a Simple Linear Programming Problem.

that the maximum *feasible* value of the objective function occurs at a vertex of the polygon (15/7,-8/7). In fact, it will always happen (even in higher dimensional space) that the objective function for a linear programming problem will assume its extreme value at a vertex of the space of feasible solutions. All linear programming algorithms employ the strategy of examining the objective function only at vertices of the region of feasible solutions. The algorithm moves from vertex to vertex in such a way that the objective function value at the successor vertex is at least as great as at the predecessor vertex. In a finite number of steps the optimum can be found.

The algorithms required to solve a linear programming problem are too complicated to describe briefly. Detailed descriptions can be found in many texts, among the more readable being Gass,[6] Hadley,[7] and Dantzig.[8]

Nonlinear Programming

In the nonlinear programming problem, the objective function or one or more of the constraints assumes a nonlinear form; the constraint functions are normally defined over continuous ranges of the independent variables. When the objective function to be maximized is subject to equality constraints and both the objective function and the constraint function can be differentiated, it may be possible to use one of the techniques of variational mathematics such Lagrange multipliers. Unfortunately, in most engineering design problems, constraints are of the inequality form and the objective function

[6] Saul I. Gass, *Linear Programming Methods and Applications*, 2nd ed., McGraw-Hill Book Company, 1964.

[7] G. Hadley, *Linear Programming*, Addison-Wesley, Reading, Mass. 1962.

[8] George B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey, 1963.

is seldom easily differentiable. Consequently, Lagrange multipliers have only limited utility for the more difficult engineering design problems.

So-called simplicial methods have met with a certain success. These all transform the nonlinear programming problem into an iterative sequence of linear programming problems. One of these approaches, known as the cutting plane method, is described briefly below. Still other approaches to the nonlinear programming problem involve modifications of methods for unconstrained functions of many variables, such as those already discussed. Two which appear to have achieved a substantial acceptance are the gradient projection method, a modification of the method of steepest ascent, and the multiple gradient summation method which is a modification of the direct search of Hooke and Jeeves.

Cutting Plane Method: In this method[9] the general nonlinear programming problem maximize $f(\bar{x})$ subject to

$$g_i(\bar{x}) \leq 0, \ i = 1, \ldots, m$$
$$\bar{x} = [x_1, x_2, \ldots, x_n]$$

is rewritten as a problem with a linear objective function and nonlinear constraint by adding one new variable and one new constraint,

$$g_{m+1} = x_{n+1} - f(\bar{x})$$

Then the problem is: maximize x_{n+1} subject to

$$g_i\ (\bar{x}) \leq 0\ , \ i = 1, \ldots, m$$
$$g_{m+1} \leq 0$$

In the cutting plane approach, each constraint function, $g_i(\bar{x})$, is replaced by a linear approximation obtained by expanding it about some starting point, $\bar{x}^0$, in a Taylor series and dropping all terms of order two and above. The nonlinear problem is then solved by first solving: maximize x_{n+1} subject to

$$g_i(\bar{x}^0) + g_i\ (\bar{x}^0)\ (\bar{x} - \bar{x}^0) \leq 0$$
$$i = 1,2, \ldots m+1$$

The solution, $\bar{x}^*$, of this *linear* programming problem (which may or may not fail to satisfy all of the original constraints) can be used as the base for new Taylor series expansions of the constraints, creating a new linear programming problem to be solved. When the solutions $\bar{x}^*_k$ and $\bar{x}^*_{k+1}$ for two successive iterations differ by less than some acceptable amount and $\bar{x}^*_{k+1}$ satisfies the original constraints, a solution to the nonlinear problem has been found.

METHODS WHICH EXPLOIT SYSTEM OR INFORMATION FLOW STRUCTURE

The methods described earlier have an essentially sequential point-to-point character; all the design variables are selected simultaneously for a given point, i. e., if there are n design variables, then the problem becomes one of searching for the optimum in n-dimensional space. It frequently happens that the structure of the system can be exploited to allow the optimization problem to be reformulated as a stage-wise decision process, i. e., formulated in such a way that computation of the optimum solution takes on a serial character involving at each stage or step in the process the variation of no more than a very few of the n design variables at a time. One powerful solution approach to such problems which may involve a very large number of design variables is called dynamic programming.

The stages are usually numbered in reverse order to the direction of information or material flow (see Figure 23).

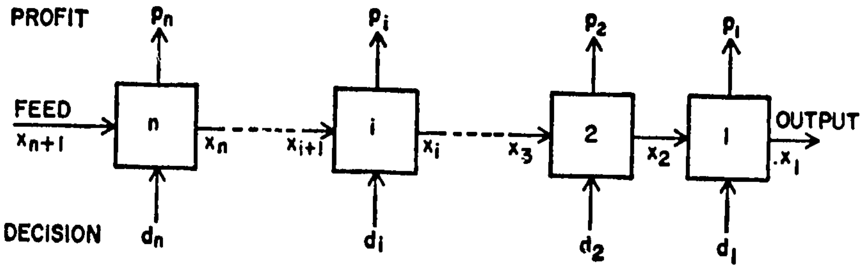


Figure 23: Dynamic Programming.

The input or state variable to the i'th stage is the output state variable for the (i+1)'th stage x_{i+1}. (x_{i+1} might be a vector of state variables $\bar{x}_{i+1}$.) A decision or design variable d_i (or vector of decisions $\bar{d}_i$) is specified for the i'th stage. For each stage there are two functional relationships. The first, denoted h_i describes the output state variable for stage i, x_i in terms of the input state variable x_{i+1} and decision made d_i. The second, denoted p_i, describes the profit (negative if a cost) associated with the operation of the stage. The object of the dynamic programming algorithm is to determine how to operate the system (i. e., what decisions $d_1, d_2, \ldots, d_n$ should be made) to yield a maximum overall profit.

Operation of Stage $\quad x_i = h_i(x_{i+1}, d_i)$

Stage Objective
Function (e.g. Profit) $\quad p_i = g_i(x_{i+1}, d_i)$

Optimal Cumulative Object Function:

$$f_i(x_{i+1}) = \max_{d_i}.\left(p_i(x_{i+1}, d_i) + f_{i-1}(x_i)\right)$$

The procedure, which is a bit too complicated to detail here, is based on Bellman's principle of optimality, which states:

> "An optimal policy has the property that, whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

[9] J. E. Kelley, Jr., "The Cutting Plane Method for Solving Convex Programs," *J. SIAM,* 8, pp. 703-712, 1960.

As a consequence of this principle, it is possible to optimize a serial system using a tail-to-head strategy, first examining a one-stage process, then a two-stage process, etc.

Provided the program can be appropriately formulated, an apparently n-dimensional problem can be solved as a sequence of m-dimensional problems where $(m \ll n)$; in some cases the n-dimensional problem can be reduced to a series of n one-dimensional problems. The number of computations required can be reduced by several orders of magnitude over that required by standard optimum seeking methods when the dynamic programming approach is applicable. Even so, for problems with a large number of state and decision variables, at each stage the computations problem can become quite formidable.

During the past year or two, and inspired primarily by the success of the dynamic programming concept, there have been several studies involving the decomposition of macrosystems, i.e., large and complex systems containing many components or subsystems. The object of the decomposition schemes is to allow effective suboptimization of individual subsystems, using virtually any of the various mathematical optimization techniques, both simultaneous and sequential, to reduce a problem of hopelessly large dimensionality to one of computable proportions. A recent paper by Wilde[10] is particularly well written.

DESIGN PROBLEMS

During the last four weeks of the nine-week summer program of the Project on Computers in Engineering Design Education, each of the participating faculty members generated, solved, and documented one or more design problems which he felt suitable for use in his design course. Most of these problems are included in the disciplinary volumes of the final report of the project. Each problem is stated as it might be presented to a student in an engineering design course and detailed descriptions of the solution methods, computer programs and computer results are included with each problem.

Most of the faculty participants attempted to use one or more of the mathematical optimization methods of the previous pages to produce a design that was "best" in some sense. Table VII lists the volume number and general field of interest for those problems involving some optimization strategy. The number of the problem used in the reports, its title, the type of optimization method, the nature of the objective function, and the design variables involved are listed under each. Those interested in details should refer to the appropriate volume of the project report.[11]

TABLE VII

Engineering Design Problems Using Optimization Methods

(Note): The format adopted is:
a. Optimization method.
b. Objective function.
c. Design variables.

Volume II — Chemical Engineering

1. Economic Optimum Conditions for a Staged Separation Process
   a. Univariate Search
   b. Rate of return =
   $$\frac{(\text{Income - Product Cost}) - \text{Capital Investment}}{\text{Capital Investment}}$$
   c. XD = Product purity, MF = Multiple of minimum reflux, FR=Fractional recovery

4. Design of a Mixed-Suspension, Mixed-Product Crystallizer
   a. Exhaustive Search
   b. Rate of return $= \dfrac{\text{Income-Total Cost}}{\text{Capital Investment}}$
   c. $\ell =_{\text{avg}}$ Average crystal size

Volume III—Civil Engineering

6. Optimal Design of Two-Way Reinforced Concrete Slabs
   a. Exhaustive Search
   b. Either weight of slab or cost of slab
   c. t = Slab thickness

7. Decision Making in Building Planning
   a. Linear Programming, Simplex Method
   b. Cost of apartment complex
   c. Number of apartments built for each price range and size

8. Optimization of a Two-Span Cover-Plated Steel Beam
   a. Nonlinear Programming, Cutting Plane Method
   b. Weight of the beam
   c. b_f = Flange width, t_f = Flange thickness, d_w = Web depth, t_w = Web thickness, ℓ_1 = Left span length, ℓ_2 = Right span length, b_p = Cover plate width, t_p = Cover plate thickness, etc.

Volume I — Summary
Volume II — Chemical Engineering
Volume III — Civil Engineering
Volume IV — Electrical Engineering
Volume V — Industrial Engineering
Volume VI — Mechanical Engineering
Final Report, Project on Use of Computers and Mathematical Optimization Techniques in Engineering Design, D. L. Katz, B. Carnahan, R. Douty, E. McMahon, N. Hauser, J. Zimmerman, and W. Seider, March 1966, University of Michigan (800 pages).

[10] D. J. Wilde, "Strategies for Optimizing Macrosystems," *Chem. Eng. Progress, 61*, No. 3, pp. 86-93, March 1965.

[11] *Computers in Engineering Design Education.*

9. Design of Rectangular Tied Concrete Columns
   a. Exhaustive Search
   b. Cost of columns
   c. Size and number of longitudinal bars, size and spreading of lateral ties

10. Optimal Design of Structural Steel Framing for Tier-Type Buildings
   a. Dynamic Programming
   b. Cost of a tier-type building
   c. Joist spacing, joist span, number of beam panels, number of girder panels, column specifications

11. Minimum Weight Design of a Linear Elastic Column
   a. Univariate Search
   b. Volume of material
   c. Column radius, length, and wall thickness

12. Optimal Design of a Timber Warehouse Floor
   a. Dynamic Programming
   b. Cost of a warehouse floor
   c. Joist spacing, floor material, joist material, beam material

13. Optimization in Site Selection
   a. Constrained Gradient Search
   b. Cost of a static missile test stand including utility and road construction
   c. Distance of missile stand from water sources, drainage pond, barricades, and roads

14. Design Force for Bracing in a Deep Vertical Cut
   a. Direct Search
   b. Bracing design force
   c. Horizontal position of logarithmic spiral cut

15. Optimal Design of a Wide-Flange Beam Using Standard Rolled Shapes
   a. Exhaustive Search
   b. Weight of the beam
   c. Beam dimensions

Volume IV — Electrical Engineering

17. Design of a Ferromagnetic Circuit
   a. Dichotomous Search
   b. Mass of the circuit
   c. B = Width of leg 3, K = Width ratio-leg 2/leg 3, etc.

20. Class C Amplifier Design
   a. Gradient Search
   b. A trade-off function including input and output signal powers, and plate efficiency
   c. Plate supply voltage, grid supply voltage, minimum plate voltage

Volume V—Industrial Engineering

22. A Dynamic Programming Problem for the Assignment of Tolerances
   a. Dynamic Programming
   b. Cost per combination within overall tolerance specified
   c. Component (stage) variances

25. Optimal Assignment of Machines to an Operator
   a. Exhaustive Search
   b. Cost per piece manufactured
   c. Machine cycle time, operator idle time, machine idle time

26. A Dynamic Programming Approach to an Inventory Problem
   a. Dynamic Programming
   b. Cost of maintaining inventory
   c. Schedule for purchasing

Volume VI — Mechanical Engineering

29. Rocket Nozzle Design
   a. Dichotomous Search
   b. Weight of nozzle wall
   c. Wall thickness, materials of construction

30. Gear-Ratio Determination by Computer-Assisted Search
   a. Cartesian Walk
   b. Precision of gear-ratio determination
   c. Number of teeth

31. Optimization of a Pneumatic Linear Actuator
   a. Direct Search
   b. Operating cost (compressed air usage) usage)
   c. 11 variables including working area, stroke length, spring rate, working pressure, opening area, etc.

32. Ejection Actuator
   a. Monte Carlo Methods
   b. Reliability of actuator system
   c. Resistance force

33. Determination of Optimum Pipeline Route
   a. Dynamic Programming
   b. Cost of a pipeline route
   c. Number of sites, pump, and pipe sizes

34. Cam Design of Minimum Size Based on Bearing Stress Limitations
   a. Direct Search
   b. Contact stress
   c. Prime circle radius

35. Optimum Reheat Pressure for a Steam Power Cycle
   a. Fibonacci Lattice Search
   b. Heat cycle efficiency
   c. Extraction pressure

THE SYSTEM APPROACH TO COMPUTER-AIDED DESIGN: USE OF PROBLEM-ORIENTED LANGUAGES

R. L. MOTARD
University of Houston
Houston, Tex.

Modern engineering design continues to expand in complexity and depth under the impact of computers. The use of computers in the control of chemical plant operations places an additional burden on the engineering analysis of complete process system. Adaptation is most certainly taking place in engineering curricula to meet these requirements. A recent report in a survey of accredited engineering programs indicates an exponential growth in course changes. In some respects change and adaptation are addicting and the prevailing atmosphere in most academic institutions now favors continued evolution in the pursuit of excellence.

No doubt modern engineering design demands higher sophistication in mathematical skills, more intimate contact with computers and some attention to mathematical optimization; with further emphasis on the systems approach, we complete the picture. Many engineering schools are struggling with the expansion of these topics in the undergraduate curricula. Hopefully these developments will add to the young engineer's skill in design but, as we have come to appreciate, something more is needed, namely, a broader experience with open-ended design problem situations than is generally available to the undergraduate. Perhaps it has already been said enough, but you will forgive me if I repeat, that creative design is both synthesis and analysis.[1] The something more that I referred to above is the missing component that mathematics, computers and systems do not confer upon the student. We need a component that adds synthetical vision, creativity, intuition and inductive skill to the engineer's mental equipment.

Any mental process that can be developed and described as a step-by-step procedure is analysis. It is the stuff of scientific verification and validation. Analysis formalizes for the benefit of scientific or engineering fraternity the structure of a solution and its relation to sound principles. Synthesis, too, must be grounded in scientific principles that have been thoroughly assimilated, but it relates differently to these disciplines. We must not confuse synthesis with analysis and vice versa, since they are distinctive modes of thought. Although it is an oversimplification, we can say that synthesis is largely a subjective process and analysis is largely objective. By this I mean that analysis, following the formal logic of language and mathematical symbolism, is an external medium of expression useful for the recording of completed conceptions and for the social exchange of scientific information. Synthesis, on the other hand, depends on the infralogical processes of the mind and there are now some philosophers of science [2, 3] who maintain that such processes are not couched in the external and formal logical language but depend on unknown mental symbolism more closely related to unknown modelling and heuristic maneuvers and relying on intuitive notions of analogy, symmetry, limits and visualization.[4] I would hazard a guess that many mental models in design synthesis spring from physiological and visual analogs of the system and its elements.

The proper academic medium for the development of synthetic skill in design must involve direct experience with total systems since every design must always begin at this stage. Every designer or problem-solver hungers for a perception of his total problem. Systems science and engineering offer him some guidance in structuring and classifying the concepts which have been refined in past design work with total systems. However, such terms as control, optimization, transfer function, feedback, time delay, dynamic response and communication channel are almost meaningless to the average undergraduate student. Yet it is precisely such concepts or their analogs which he must consider in modern design situations. We have trained him rather well in analysis and he usually enters a senior design project very much task-oriented. He is not prepared to innovate and create systems solutions to open-ended problem statements. If we borrow some more of Bruner's notions,[5] our aver-

[1] Motard, R. L., "SYSTEMS ENGINEERING: Engineering Come of Age and Its Academic Image," *Journal of Engineering Education*, vol. 56, p. 198 (1966).

[2] Moles, A., *La Creation Scientifique*, Editions Rene Kister, Geneva, 1957.

[3] Bronowski, J., "The Logic of the Mind," *American Scientist, 54*, 1 (1966).

[4] Bruner, Jerome S., *The Process of Education*, Random House (Vintage Books), New York, 1960.

[5] Bruner, Jerome, S., ibid.

age student has been brought up on the reward and punishment system and has seldom experienced complete absorption, intensive curiosity and the lure of discovery.

What is proposed, therefore, is a laboratory experience with total systems in the student's field of interest which are intended to heighten his intuitive grasp of systems and to stimulate his creativity before he is asked to produce a new design. Many other approaches are possible, but this one lends itself well, I believe, to upper-division design courses when the student has already mastered many analytical procedures and understands the elements of his systems reasonably well. The systems in this case are computer simulations from which the student can extract information quite simply and easily, yet they are of such complexity that he can never be quite assured of a total understanding. The student should experience the same sense of wonder, mystery and discovery that he would find in exploring the operation of the actual hardware. His experience with the system would consist of two phases: the first or familiarization phase wherein the optimum operation of an existing structure is sought; and the second phase in which a restructuring of the system is allowed. Such a problem should then parallel real-world experience in design but with the added incentive of "instant" experience in allowing the computer to perform most of the analysis which the student follows vicariously.

PROBLEM-ORIENTED LANGUAGE

The simulation of complex systems in itself is not a task for the undergraduate student. The problems are simply too vast and detailed to be done as classroom projects. It is more practical to provide a computer-executive system which the user can have easy access to and structure to his own needs. The language of communication with the computer is problem-oriented in that it is especially apt in describing systems common to a given engineering discipline. The executive system used in the example discussed below is a major revision of the PACER executive system which was originally developed by Professor Paul T. Shannon of Dartmouth College and is now implemented in the MAD language on the University of Michigan computer-operating system. The MAD language lends itself very well to this type of development because of its great sophistication and flexibility, its information-processing ability and the simplicity of its input.

This chemical process engineering language is designed to fit the structure of an ordinary chemical plant in which a set of streams of gas, liquid or solid materials are heated, separated, compressed or transformed chemically in an interconnecting matrix of process units. The executive system performs heat and material balances on the simulated process and calls upon the process-unit calculations subroutines to carry out the chemical or physical transformations. All of these are provided for the student. The key to the great simplicity of use of the language is the limited amount of data to be supplied to carry out a simulation. A comprehensive thermodynamic-data subroutine provides all the necessary phase equilibrium and energy data. Only hydrocarbon systems are currently programmed.

The details of the subroutines and executive system are given in the Chemical Engineering Section of the Report of the University of Michigan Project on the Use of Computers in Engineering Design. In brief, the basic input data for a simulation are shown in Table I.

TABLE I
Basic Input to the Chemical Process Engineering System

1. Job description for output identification.
2. Control constants.
   Number of components per stream.
   Component identification and names.
   New job or continuation.
   Sizes of principal data matrices.
   Process unit numbers in recycle loops.
   Tolerance.
3. Process matrix, one row vector for each unit.
4. Stream composition matrix for each process feed stream.
5. Equipment parameters matrix, one row vector for each unit.
6. Process control and convergence forcing information.

The structure of the process and stream matrix row vectors are shown in Table II and the flowchart for the executive system in Figure 1. A typical coding for a unit and its extension are shown in Figure 2. Further details of the system operation are discussed in the example problem and results.

TABLE II
Structure of Process and Stream Matrix Row Vectors

KPM (I, 1) = Unit number, I	SN (J, 1) = Stream number, J
KPM (I, 2) = Unit SR name	SN (J, 2) = Stream type
KPM (I, 3) = External name	SN (J, 3) = Enthalpy
KPM (I, 4) = Input stream	SN (J, 4) = Temperature
KPM (I, 5) = Numbers (positive)	SN (J, 5) = Pressure
	SN (J, 6) = Total moles
	SN (J, 7) = Moles component 1
KPM (I, -) = Output stream numbers (negative) etc.	SN (J, 8) = Moles component 2 etc.

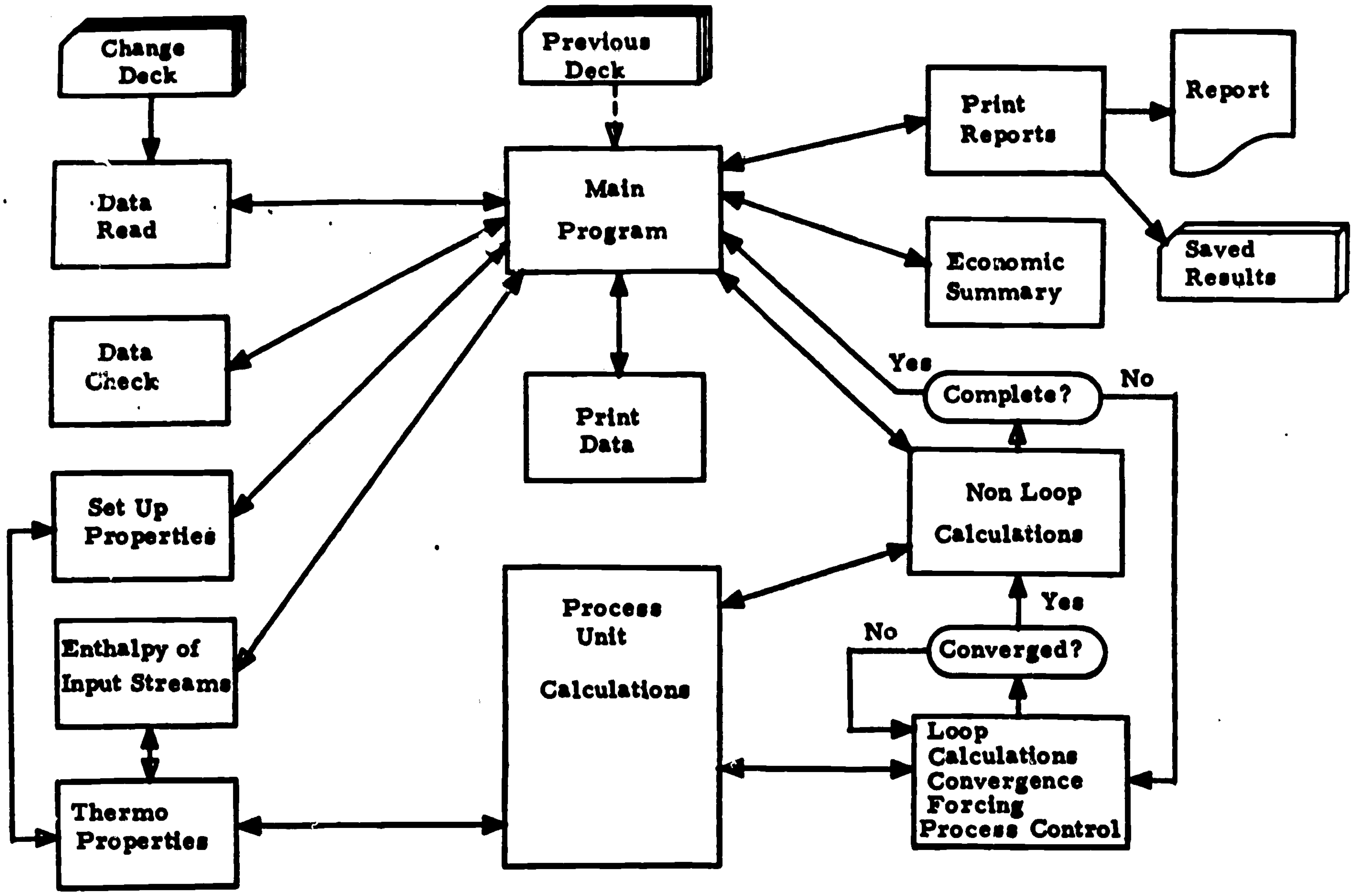


FIGURE 1

Flowchart of Executive System

KPM = Process matrix
SN = Stream matrix
ENC = Equipment matrix

KPM(8, 1) = 8, $EXTRA4$, $DIST2$,
1, 3, 4, -2, -5

SN(1, 1) = 1., 1., 0., 650., 150., 45.4., ...,
ENC(8, 1) = 8., 10., 5., 1., 10., ...,

ENC(4, 1) = 4., ...,
KPM(4, 1) = 4, $UNAME7$, $E1$, 2, -6
KPM(5, 1) = 5, $UNAME4$, $V1$, 6, -8, -9
KPM(6, 1) = 6, $UNAME1$, $D1V$, 9, -3, -10,
KPM(7, 1) = 7, $UNAME7$, $E2$, 5, 12, -7, -13
KPM(9, 1) = 9, $UNAME4$, $V2$, 7, -4, -11,

ENC(5, 1) = 5., ...,
ENC(6, 1) = 6., ...,
ENC(7, 1) = 7., ...,
ENC(9, 1) = 9., ...,

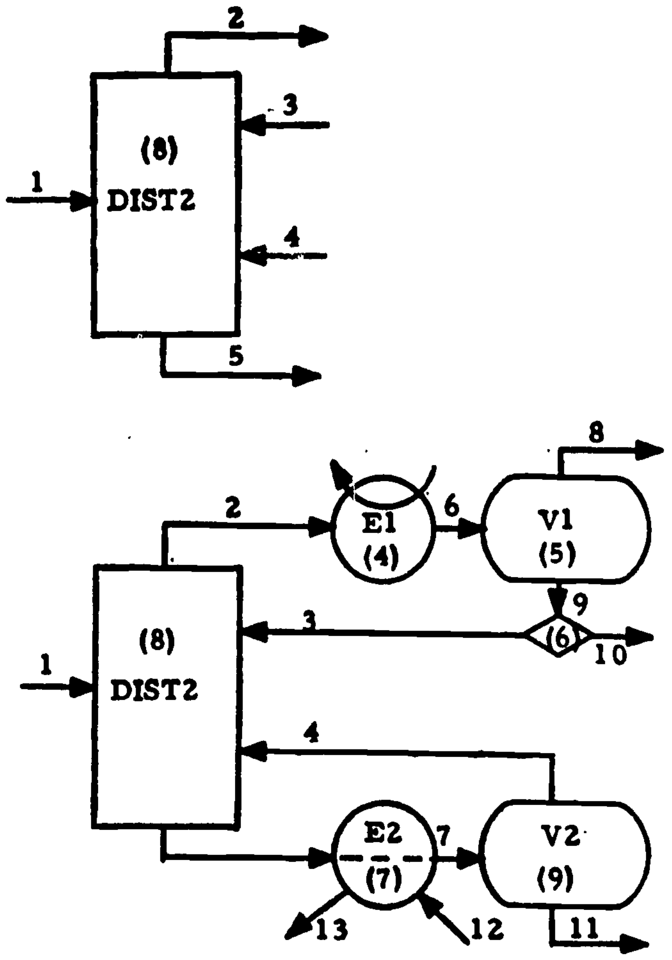


FIGURE 2.
Use of Problem Oriented Language

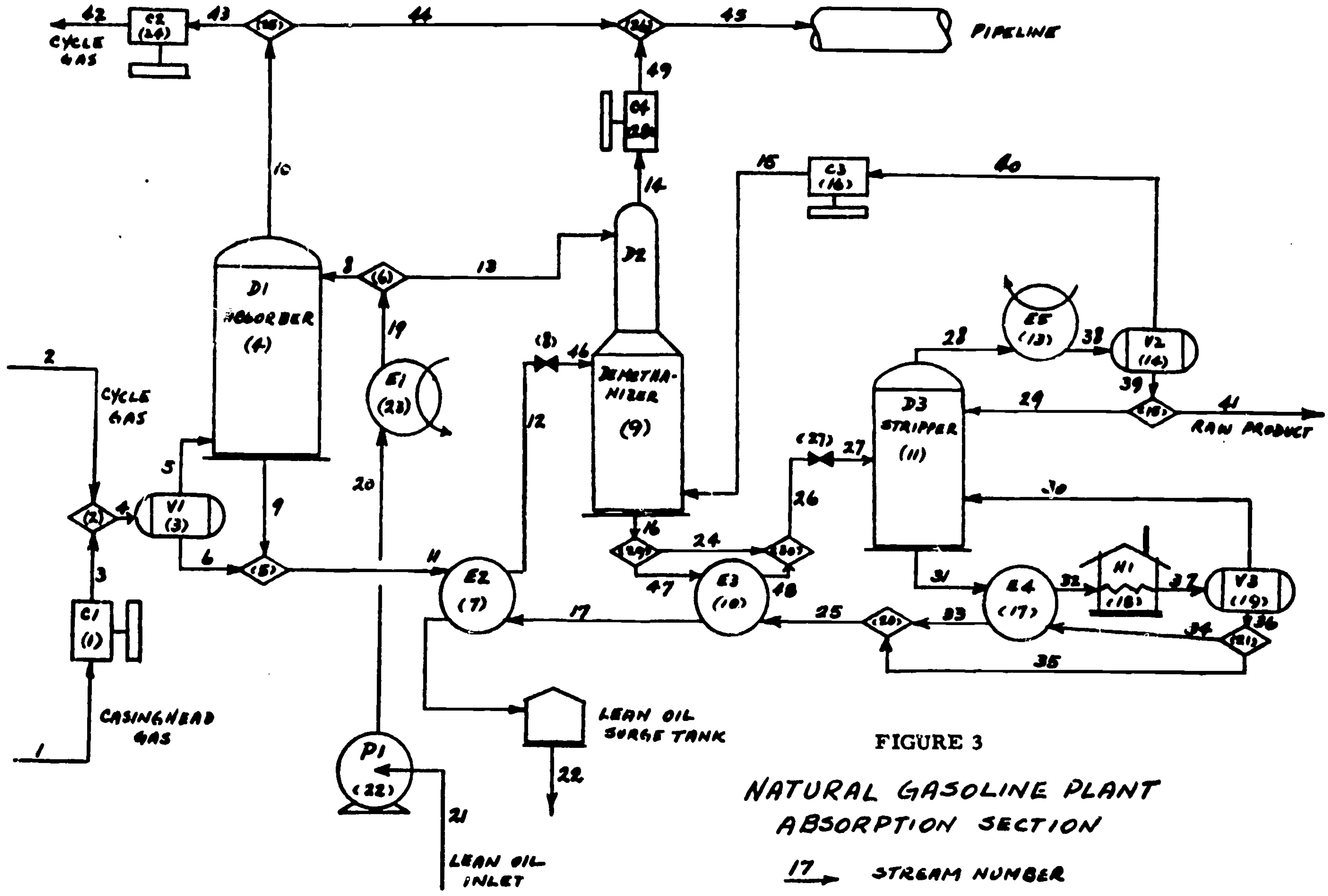


FIGURE 3

NATURAL GASOLINE PLANT
ABSORPTION SECTION

17 → STREAM NUMBER

(5) UNIT OR EQUIPMENT NUMBER

NOTE: (12) AND 23, 7, 18 ARE NOT USED

EXAMPLE PROBLEM

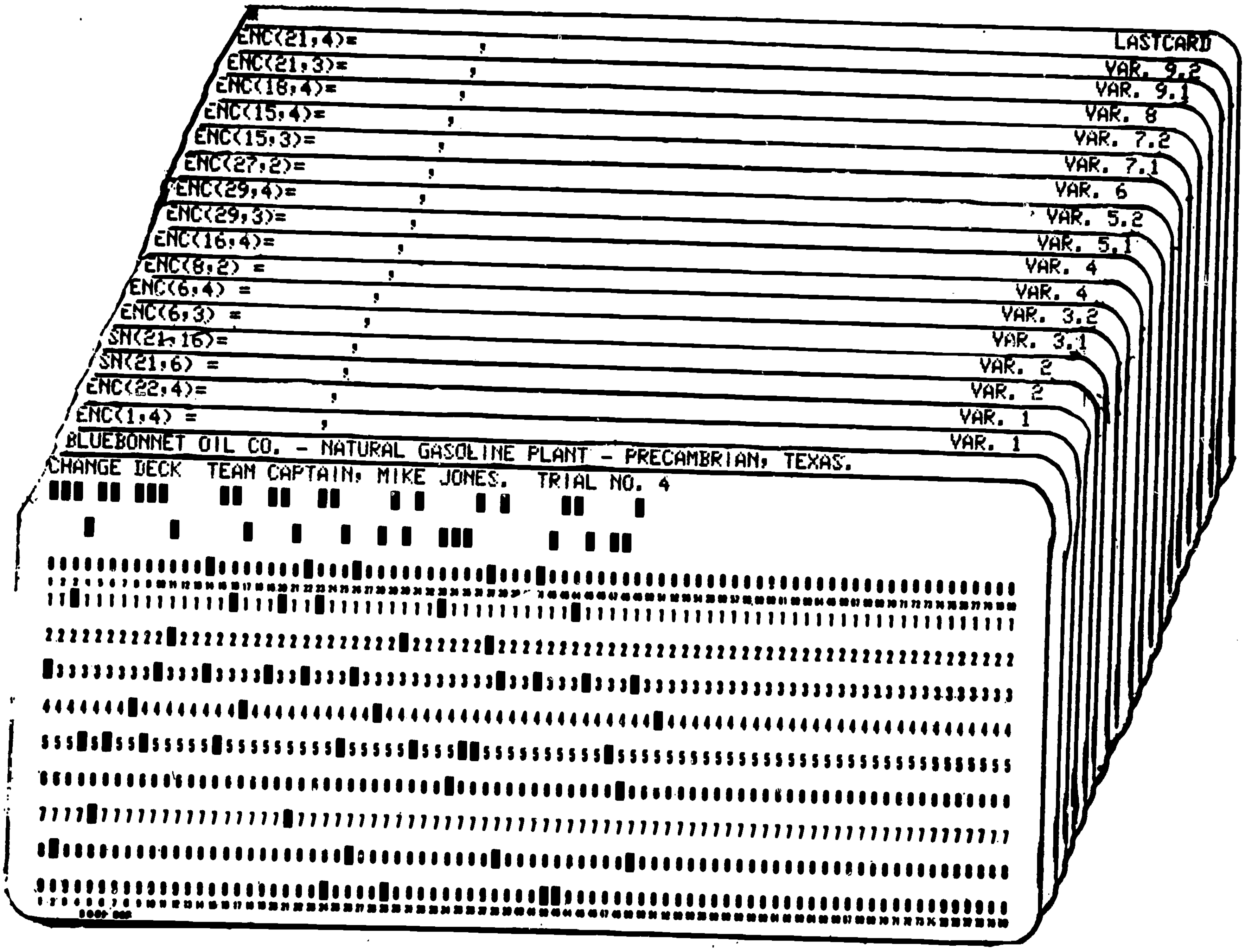
The process chosen for classroom experimentation is a natural gasoline plant. The process flowsheet is shown in Figure 3.

Sufficient flexibility was allowed in the structuring of the process to permit the manipulation of nine basic variables. Four of the basic variables involve stream splits and require the specification of the joint stream ratios. The object of the first phase of the problem, as previously stated, is to optimize the economic return from the existing plant. Each team of students, four or five men to a team, is supplied with a coding form and a change deck of variable cards. Only the variable cards for those variables being altered need to be submitted for any run, along with a binary deck representing the computer output of previous results from a past run. Using a previous output as a starting case reduces the computer time requirements.

Each team is allowed about ten "plays" or runs, and any set of the basic variables may be varied from run to run.

DISCUSSION OF RESULTS

Output from the program provides stream properties for all streams in the process, a total of 49, and an economic summary which can be analyzed to determine the next stage in the cooling operation. (See Figs. 4 and 5.) No guidance is given to the students regarding any formal optimization procedures and this must be done externally by the team. It is anticipated that the optimization will be done heuristically by most students. No classroom experience can be reported since the students are only beginning to use the system. Perhaps the most disturbing aspect of the whole concept is the tremendous amount of preparatory work by the instructor and the great demands on the use of the computer. The present project represents at least four man-months of instructor time and each run takes a minimum of fifteen minutes of IBM 7094 time. For the entire first phase of the project it is estimated that the four teams of students will need ten hours of computer time. Very few universities could afford to support many such projects at the present levels of computing facility. Undoubtedly any extension of educational effort in the direction of more complex computer executive systems will require more staff support for instruction, since no individual faculty member can afford to carry the entire burden.



```
ENC(21,4)=          ,                                  LASTCARD
ENC(21,3)=          ,                                  VAR. 9.2
ENC(18,4)=          ,                                  VAR. 9.1
ENC(15,4)=          ,                                  VAR. 8
ENC(15,3)=          ,                                  VAR. 7.2
ENC(27,2)=          ,                                  VAR. 7.1
ENC(29,4)=          ,                                  VAR. 6
ENC(29,3)=          ,                                  VAR. 5.2
ENC(16,4)=          ,                                  VAR. 5.1
ENC(8,2) =          ,                                  VAR. 4
ENC(6,4) =          ,                                  VAR. 4
ENC(6,3) =          ,                                  VAR. 3.2
SN(21,16)=          ,                                  VAR. 3.1
SN(21,6) =          ,                                  VAR. 2
ENC(22,4)=          ,                                  VAR. 2
ENC(1,4) =          ,                                  VAR. 1
BLUEBONNET OIL CO. - NATURAL GASOLINE PLANT - PRECAMBRIAN, TEXAS.    VAR. 1
CHANGE DECK   TEAM CAPTAIN, MIKE JONES.   TRIAL NO. 4
```

FIGURE 4

Change Deck of Variable Cards

STREAM NUMBER	39	40	41	42	43
EQUIP. CONXION	FR 14 TO 15	FR 14 TO 16	FR 15 TO 0	FR 24 TO 0	FR 25 TO 24
VAPOR FRACTION	.0000	1.0000	.0000	1.0000	1.0000
ENTHALPY, BTU	4554.6212	36992.6064	2277.3106	77095.7373	85780.6436
TEMPERATURE, R	569.2714	569.2714	569.2714	646.3154	646.3154
PRESSURE, PSIA	80.0000	80.0000	80.0000	3000.0000	1000.0000
			COMPOSITION, LB-MOLES/HOUR		
METHANE	.0057	.5133	.0028	15.1846	15.1846
ETHANE	.0324	.7322	.0162	.4345	.4345
PROPANE	.0965	.7295	.0483	.2414	.2414
I - BUTANE	.2410	.7855	.1205	.0222	.0222
N - BUTANE	.3162	.7721	.1581	.0029	.0029
N - PENTANE	.1876	.1609	.0938	.0000	.0000
N - HEXANE	.1387	.0433	.0693	.0000	.0000
N - HEPTANE	.0893	.0110	.0446	.0000	.0000
N - OCTANE	.0420	.0020	.0210	.0000	.0000
TETRADECAN - LO	.0125	.0000	.0063	.0000	.0000
TOTAL	1.1619	3.7498	.5809	15.8857	15.8857

FIGURE 5

STREAM PROPERTIES OUTPUT FORMAT

UTILITIES CONSUMPTION

	WATER GPH	STEAM LBS/HR	ELECTRIVITY KWH	FUEL GAS SCFH
UNIT COSTS	.00003	.00040	.01500	.00035
EQUIP. NO.				
1	177	0	.0	42
13	2067	0	.0	0
16	—0	0	.0	0
18	0	0	.0	1048
22	0	0	4	0
23	1419	0	.0	0
24	173	0	.0	45
28	70	0	.0	25
TOTAL USE	3905	0	4	1160
	. 12	.00	.06	.41

TOTAL COST .58

PRODUCT VALUES, STREAM NO. 41

	MOLES/HR	UNIT VALUE	VALUE
METHANE	.00285	.0000	.00
ETHANE	.01619	.0967	.00
PROPANE	.04825	.2088	.01
I - BUTANE	.12052	.3585	.04
N -- BUTANE	.15808	.4960	.08
N - PENTANE	.09382	.8244	.08
N - HEXANE	.06933	.9354	.06
N - HEPTANE	.04463	1.0494	.05
N - OCTANE	.02099	1.1700	.02
TETRADECANE - LO	.00626	.0000	.00
TOTAL VALUE, DOLLARS			.35

FIGURE 6

Economic Summary for Gasoline Plant Simulation

COMPUTERS IN ENGINEERING DESIGN AT UNIVERSITIES

SECTION D

CHAIRMAN: L. A. SCHMIT, JR., *Prof. of Structures*,
Engineering Division, Case Institute of Technology,
Cleveland, Ohio

COMPUTER-AIDED DESIGN

ALLEN B. ROSENSTEIN
University of California
Los Angeles, Calif.

Foreword: A brief summary is appended of the courses with a computer-design orientation given or planned by the Engineering Department of UCLA (Appendix I). Required undergraduate core curriculum courses are listed along with selected graduate courses. Course outlines or any additional information are available upon request.

INTRODUCTION

The Commission on Engineering Education is to be thanked for organizing a conference at this time to assess the impact of computer-aided design upon engineering education.

Unfortunately, with only a few notable exceptions, engineering educators have been very late in integrating the computer into the design process. In this day of high-speed information processing, insofar as we in engineering education emphasize manual analysis and ignore problem formulation, syntheses, machine information processing, optimization, and decision-making, we are turning out men who face early technological obsolence.

The course that I wish to describe in this paper has been offered at UCLA as a graduate seminar (Appendix II). It has been proposed as part of the preparation for a doctoral field in Engineering Design. The mass of material now becoming available leads us to believe that the course will eventually expand into two semesters with supporting offerings in optimization and simulation. It is expected that, as the material becomes more formalized, a substantial portion will find its way into the undergraduate curriculum. As this happens, the arguments over design as a discipline will become academic. I would predict that computer-aided design courses have a high probability of becoming the key part of widely required undergraduate design stems.

Our computer-aided design course is based upon several assumptions. The first assumption is that design taken in its broadest sense is a formal discipline. We take for granted the fact that the design process possesses logic, structure, and methodology. Furthermore, this process is amenable to generalization and therefore it can be described, taught, and put into practice. The morphology of the overall design process has been described by Asimow. I have called the basic elements of design, which are repeated over and over again in the design process, the Anatomy of Design.

Design has been defined as an iterative decision-making process. The computer-aided design course is built upon the assumption that once the decision rules and procedures for the design of a product have been completely formulated, all future designs using the same rules and information sources can be produced faster and better by machine than by hand. We can restate this assumption by saying that if an engineer can describe to another engineer in an unambiguous fashion the procedure necessary to design a given product, then he can also transfer the same rules to a machine.

Design problems are almost never formulated with enough equations to cover all of the parameters. An educated search procedure is usually required. However, once the rules and logic for these procedures are formalized, they can be better performed by machine. The course, therefore, looks for the generalized computer-aided design concepts which are applicable in all fields of engineering.

We begin by attempting to give an organized overall view of the present state of computer-aided design with some prediction of its future. The design process is reviewed and an analysis is made of the comparative present and future contributions of the engineer and the computer to each element of the design process (Appendix III).

The expectations for generalized computer-aided design procedures become apparent when we examine the major factors of the design anatomy and catalogue the available design parameters, analytical models, and optimization techniques. The limited classes of parameters to be varied in any design are identified as geometry, materials, and energy (these sometimes appearing as people and information). The analysis problem hinges upon the selection of either a lumped parameter, distributed parameter or probabilistic model. The optimization procedures available are also not extensive.

Applications to illustrate computer-aided design are organized under the general headings of:

1. Component Design.
2. System Design.

From the standpoint of the overall design process, there is no difference between the design of components and the design of systems. At present we think we see a subtle but significant distinction between the parameters of components and those of systems.

We generally find the design parameters of components and systems can be compared as follows:

Component Parameters	System Parameters
Materials	Components
Geometry	Topology

The distinction between the design procedures for components and systems seems to lie primarily in the number of independent parameters that the engineer or the computer is called upon to handle. The number of independent parameters that are perturbated in the design of a component is relatively small and an optimum solution can be sought.

COMPONENT DESIGN BY COMPUTER

Component design is typified by the direct perturbation of allowable materials and geometries. The component may intially possess a relatively large number of parameters. However, statement of the equation of constraint arising from the laws of nature, laws of society, customer specifications and the house rules of the manufacturing concern will generally reduce the number of independent variables to a more manageable number.

Today major components ranging from transformers, motors, and filter networks to turbines, foundations, and the structure of mulitistage rockets are regularly designed by computers. To illustrate a general component-design procedure, the course investigates the computer design of commercial three-phase induction motors.

Induction-motor design requires the detailing of at least 26 parameters. The equations of constraint permit a reduction to five independent parameters. One of these is a material parameter while the other four are geometric parameters. They are:

1. Damper bar material — Material.
2. End ring cross section
3. Punching diameter
4. Core length } Geometry.
5. Flux density - winding turns

The computer procedure begins with the customer's specifications. A sizing routine based upon the experience gained from past designs determines the maximum and minimum volumes to be expected. Volume considerations and standard punchings then determine the range of usable standard laminations and the maximum rotor length. With the design thus bounded, the computer then conducts a search for the optimum design as defined by the specified criterion function. Full design and manufacturing data are read out.

The gross similarities of component design procedures make the prospects for fully generalized component design routine appear excellent. The procedures used to design an induction motor are applicable to other components, such as a structural member.

COMPUTER-AIDED SYSTEM DESIGN

System design is typified by large numbers of independent parameters. The amount of data to be processed becomes so massive that geometry and materials can no longer be considered in the equations. The components of the system are then represented by their through-and-across relationship (transfer or influence coefficient). The interconnection of the components is given by topology. The system design consists of perturbation of topology and components.

Although computer-aided system design has made significant advances in recent years, and a great deal of systems analysis can be done only by means of a computer, the surface of this major application has only been scratched. The present state of the computer-aided system design art resolves into a man-machine partnership. The engineer will select a topology and initial components which will hopefully be capable of meeting the system requirements. The computer will then perturbate the components of the topology to perform a quasi-system optimization. The resultant performance and component values are then read back to the engineer, who can either accept them or suggest a new topology to the computer. The dialogue between man and machine continues until the engineer has exhausted his repertoire and selected the best of the quasi-optimized alternatives.

The computer-aided design course introduces a system designed by studying the current advances in electronic circuit synthesis with computers. The annual expenditure of engineering man-hours on the design of electronic systems has spurred the extensive application of computers to this area. I have been told that a substantial percentage of the Minute Man Missile-Guidance system was produced with the assistance of computer-aided circuit design.

Computer routines now exist which only require the engineer to spell out the topology by identifying system nodes, the connections between nodes, and the type of components making the connections. The computer then converts this input data into the appropriate matrix equations. The equations are solved for any combination of inputs and outputs to give the d.c. performance, the a.c. performance and the transient behavior. The components may be both active and passive, and may also be linear or nonlinear.

In addition to the performance to be expected from the computer component values, the computer will analyze effects of parameter shift. The "worst case" system performance is then provided with all components taken at their worst extremes for temperature, aging, and manufacturing tolerances. Since "worst case" for a large system is seldom realistic, Monte Carlo routines are also available to give the expected performance spreads of production lots.

Electronic circuit design may at first glance seem to be highly specialized. However, the nonlinear, lumped-parameter model of the electronic circuit has broad applicability to other physical systems.

COMPUTER-AIDED PRELIMINARY DESIGN

A complicated process often can be profitably categorized from more than one viewpoint. Professor Asimow has listed the sequential steps in the Morphology of Design as determination of need, feasibility study, preliminary design, detailed design, production, distribution, consumption, and retirement. While repetitive detailed design is obviously within the province of the computer, preliminary design in most industrial organizations usually requires their best engineering talent. Yet even here the computer has become more than a big adding machine. The following quotations are taken from a paper written by engineers of a leading aero-space concern.[1] They are commenting upon their corporation's use of computers for the preliminary design of multistage rockets, but they also document the applicability of computer routines to the design of other engineering structures such as bridges or domes. They report as follows:

> "An example of the computer-aided design concept is the NAA Launch Vehicle Weight Synthesis Program which was developed for the preliminary design weight synthesis of multistage launch vehicles. The program is initiated with computer inputs relating to performance, specifications, and design criteria. The program computes the weight of individual structural components, tank pressurization systems, propellant systems, and insulation systems for each stage of the vehicle configuration. Other system weights, such as propulsion, flight control, guidance, electronics and instrumentation packages are handled as input numbers. The program provides a rapid means of optimizing structural weight, including associated geometry weight penalties.
>
> "The program consists basically of a sizing subroutine, a loads analysis subroutine, a tank pressure subroutine, various stress analysis subroutines, and a detailed weight analysis subroutine which includes system analysis. In addition subroutines are included which interrelate the various analyses and accomplish the optimization of the stage mass functions and *print out* calculated numbers as well as provide *profile drawings* of the vehicle.
>
> "This program provides a means of analyzing all load conditions in a trajectory and can select minimum component design weights. A complete definition is provided automatically of the minimum weight vehicle system that can be designed with the input parameters. In addition the stiffened shell sections of the structures are described by print-outs which define the material used, skin gage, areas of stiffeners, and stiffener spacing in both the longitudinal and circumferential directions. The efficiency of this procedure is illustrated by the fact that it is possible to *synthesize* a four-stage launch vehicle system by this process in ten minutes of computer time. The same problem done by hand would literally take many man-months of detailed effort."

THE FUTURE

The studies attendant upon the development of our computer-aided design course have given us some rather definite ideas about the future of engineering. No single instrument, event, or idea within the past 200 years has had the impact upon the engineering profession, and consequently engineering education, as has that of the computer.

The physical origins of the equations it manipulates are of no particular concern to the computer. The very limited number of analytical models available represent the real world; the limited number of of criterion functions and the limited number of optimizing routines guarantee the emergence of very general automatic design programs with extensive interdisciplinary applications.

With the advent of time-sharing, the availability to even the smallest engineering office of large-scale computing facilities is no longer a question of cost and accessibility. It is now possible in Los Angeles to obtain your own console, a leased line, and a good block of time on one of the largest commercial computers for a total rental that is well under $500 a month. The only temporary impediments to industry-wide computer-aided design are those of software and the lack of engineering graduates who have been educated to use this new tool.

Competitive pressures now guarantee that the activities of the engineering profession must change drastically within the next decade. All professions are committed to the production of decisions to optimize some values in the face of some degree of uncertainty. There appears little question that the speed, accuracy, memory and decision-making capacity of the computer will cause it to assume most of the routinizable analysis and decision making in engineering design.

The role of the practicing engineer is going to become more professional. Less time will be spent on direct computation, with more effort devoted to modelling, criteria establishment, and the design of

[1] L. A. Harris, J. C. Mitchell, G. W. Morgan, "Computer-Aided Design for Civil Engineering Structures," Technology Status and Trend Symposium. Marshall Space Flight Center, Huntsville, Alabama, April 21-23, 1965.

new design procedures. Optimization procedures will be used with increasing frequency, particularly after some breakthroughs have been made in the techniques for criterion function syntheses.

Engineering schools will continue to provide a good foundation in science and analysis, but increasing time will be devoted to modelling, synthesis, and optimization. Particular attention will be directed to the interface between a design and the society that the design purports to serve. Courses in "applied humanities" are beginning to appear.

General realization that the answers to any substantial engineering problem must be a direct function of a set of criteria and values will ultimately direct a substantial concentration of engineering research effort upon the problem of value system synthesis.

The computer will have an indirect as well as a direct effect upon engineering curricula. Programming course requirements will not last any longer than the old slide rule requirement. However, lower division mathematics will be substantially reoriented to prepare students to think in computer terms and to obtain the most efficient use of the increase in availability of computer time and routines. Today we see a continuous increase in information which the engineer must handle, along with an increasing ability of the computer to process information and make decisions. This combination insures that required courses in information-processing will become as commonplace in engineering curricula as are their present-day counterparts in energy-processing.

Room for these new courses in computer-aided design and information-processing will be found through more efficient arrangements of our present so-called engineering science courses. An intensive study of upper-division engineering courses at UCLA found that, with few exceptions, there was no *new* science introduced in the engineering science courses. All of the basic science had been introduced in lower-division physics and chemistry. The engineering science courses were, in fact, courses in analysis. A typical example would be circuit theory in which the ideas of inductance, capacitance, and resistance learned in physics are employed to extend the student's ability to manipulate differential equations.

The industrial prospects for computer-aided system design are particularly exciting. The general availability of large computers through leased lines and time-sharing will cause man-machine design teams to become commonplace. Laboratory and testing time will be drastically reduced as models and routines become more sophisticated.

Component-design computer programs will undoubtedly be created to handle larger numbers of parameters. The generalized component-design programs will in turn spawn special problem-oriented routines to facilitate communication with the machine.

The design information stored in machine memories of material characteristics, component behavior, subsystem performance, and useful topologies will become extensive for every field of engineering. National design information centers will be linked to local computers. The system-design procedure written for machines will gradually become more like that used by the engineer himself. Machines in their "off hours" will read the technical literature, analyze the papers, optimize the topologies revealed, and store the resulting system-performance data in memory (after perhaps writing a letter to the editors). Concurrently, programs will be developed to allow machines to tear system specification into subsystems whose characteristics the machine can then match against the subsystem characteristics stored in memory. At this stage the computer will be able to generate its own alternative solutions to a system problem.

The major idea of the automatic factory with computerized decision-making is probably somewhat repugnant to many of us, but the elements are all available. It should be possible to deliver the customer's specification directly to the computer. The machine would then design, price all components, and prepare a bid proposal. Upon receipt of an order the computer would complete the final design, produce drawings, check inventories, order materials, program the automatic production machinery, write work orders and factory releases, establish test procedures, program the automatic check-out equipment and update the design files with the results of the proof tests. All of this would be done in addition to other routine decision-making, such as bookkeeping, payrolls, invoicing, etc.

How much of the foregoing future possibilities will become reality is, of course, a matter of conjecture. It is, however, certain that its realization and direction will be the responsibility of the young engineers we are now educating. We are beginning to realize that engineering has progressed to the point where the engineer now has the ability to solve any engineering problem which he can properly define and to which society is willing to commit sufficient resources. The design process has been well defined in recent years and its major elements revealed. It is only a matter of time before much of our present-day engineering activities will be routinized enough to become subhuman activities which we must relegate to machines.

APPENDIX I

UCLA Computer- and Design-Connected Courses
Quarter System

Engr. 6A Use of Digital Computer in Engineering. *Required Freshman Course* - 2 units.

Programming and utilization of digital computer with emphasis on the solution of engineering problems.

Engr. 16B Introduction to Design. *Required Lower Division Course* - 4 units.

Formal introduction to the design process. Problems requiring computer solution are usually included.

Engr. 100D Information-Processing Systems. *Required Upper Division Course.*

Analysis, design, and utilization of information-processing systems; representation of information, processing of information, digital systems - components, building blocks, algorithms, programming systems.

Other Required Undergraduate Experience
Upper Division Design Course and Upper Division Design Laboratories have projects which usually require some recourse to a computer.

Graduate Courses

I. Technique of System Optimization.

Theory, evaluation, and computer applications of mathematical techniques for optimization with and without constraints. Linear quadratic, and nonlinear programming, dynamic programming and stochastic processes.

II. System Simulation.

Theory and methods applicable to the simulation of a wide range of systems characterized by inherent stochastic elements.

Simulation languages including SIMSCRIPT, GPSS, and DYNAMO.
Student term projects utilizing IBM 7094.

III. Design-Automation System.

Design-automation system for digital devices. Computer design of computers.

IV. Computer-Aided Design.

Study of computer-aided engineering design. Organization of the design process, its decision points and information sources for automatic machine-processing of premature inputs (specifications) to provide full design data for any unique member of a designable family. Preliminary design, component design, systems design, and the placement and routine of problems.

A major doctoral field in Engineering Design has been proposed with several of the above courses expected to form part of the preparation.

APPENDIX II

COMPUTER-AIDED DESIGN
Course Outline

I. The Anatomy of Design.

A. Review of major elements of design process.

B. Computer as an information-processing, decision-making device.

C. Investigation of present and future roles of the engineer and the computer in the design process.

D. Component vs. system design.

II. Component Design by Computer.

A. Induction motor.

B. Transformer.

C. Waffle plate.

III. System Design with Computers.

A. Electronic circuits.

B. Structures.

IV. Placement and Routing.

V. Preliminary Design. (Example - multistage rocket).

VI. Computer-aided Design Research Frontiers.

APPENDIX III

Anatomy of Design

Operation	*Performed by*	
	Present	*Future*
I. Identification of need.	Engineer	Engineer
II. Information collection, organization and storage.	Engineer	Computer
III. Identification of system variables (inputs - outputs).	Engineer	Engineer
IV. Establish criteria and constraints. 1. Constraints: all factors which express limitations. 2. Value system and criterion function.	Engineer	Engineer + Computer Aid
V. Synthesis of alternatives.	Engineer + Computer Aid	Computer + Engineer Aid
VI. Modeling of parameters of alternatives.	Engineer	?

Parameters	Models
Geometry	Lumped parameter
Material	Distributed parameter
Energy	
People	Probabilistic

VII. Analysis - including test, evaluation and prediction.	Engineer + Computer Aid	Computer
VIII. Decision steps. Comparison of analysis of alternatives against desired results using decision rules.	Engineer + Computer Aid	Computer + Engineer Aid
IX. Optimization.	Engineer + Computer Aid	Computer
X. Communication, implementation and presentation. Paper work generation.	Engineer + Computer Aid	Computer + Engineer Aid

EBNER: *What do you see as the proper balance between the courses in an engineering curriculum and the synthesis courses?*

ROSENSTEIN: *It depends on what you plan to do with your output. Obviously, if the students are going to become engineering professors you probably don't have to teach them much synthesis. If, however, they are going to become practicing engineers, and if their main business is going to be synthesis, I think we have to strike a better balance than most of us, including my school, have struck so far. This is going to be particularly true if, indeed, we mean what we have been saying here for the last few days, that analysis is going to become a sub-human routine and that the real essence of engineering is going to be the assessment of need, the problem statement, the modeling, and probably the evaluation of the analysis and the optimization which has come out of the machine.*

As good an example as any I can think of is that of Professor Kemeny at Dartmouth who was concerned at one time about the teaching of mathematics. He picked up a sophomore examination and noticed that it required the students to derive a particular integral. Just out of curiosity he went around and asked all of his staff how they would derive that integral, and without exception they said, "Well, I'd look it up in the tables." If so much of our analysis is going to go on machines, in other words, on mechanical tables, how much time can we spend teaching the details of this? I think we should teach them something that we basically don't teach, and this lack shocks us when we analyze our engineering education. We give them a lot of workout on the analysis of problems which the professor has formulated and which have nothing to do basically with real life. Most of the time the boss says, "There's a problem over there," and if you're lucky you don't know where "there" is. But from there on in you have to formulate the problem and the analysis. You can usually get it analyzed if you can state it. In fact, this is the thing that gives me so much hope about what we're talking about, we have stated the computer-aided design problem. I am completely confident now that the details will be taken care of.

THE IMPACT OF TIME-SHARING COMPUTERS IN EDUCATION IN ENGINEERING DESIGN

PAUL T. SHANNON
Dartmouth College
Hanover, N. H.

It's a real pleasure to be here this morning. Like Jim Reswick I also brought my own computer. I owe a great deal of thanks for this to the General Electric Company and the Illinois Bell Telephone Company and I consider it an advancement in the state of the art that my remote terminal teletype is operating out in the hallway right now. I asked for it this Monday morning and it's here, operative, on Thursday. In fact, it was operational yesterday. I now have my own computer here in Chicago. I can run in Chicago; I can run at Dartmouth; I can run in New York; I can run in Phoenix—all I need is the magic set of numbers. This morning I'd like to describe very briefly the Dartmouth-General Electric Time-Sharing System developed by Professor John Kemeny and Professor Thomas Kurtz and six undergraduate mathematics students (who did 80 per cent of the programming work), to tell you a little bit about the system and how it operates, and then try to give you some appreciation of the tremendous impact this has had on our engineering education program at Dartmouth.

DARTMOUTH TIME-SHARING SYSTEM

Time-sharing computer systems are certainly the most significant development in computer technology in the last two years and will have a fantastic impact in the next five. The fundamental idea in time-sharing is the parallel servicing of multiple users, servicing them in such a fashion that they carry on a conversation with the computer so that each man believes he has his own computer. A good, concise description of the Dartmouth-General Electric 265 time-sharing system has been given by Kemeny and Tribus.[1]

"Time-Sharing Computing Systems" are just merely words until you sit down at a teletype and use the system. One of the remote terminal teletypes you see represented schematically in Figure 1 is located out in the hall. I hope as many of you as possible, during our breaks and after the sessions, will get a chance to sit down and use the system because there is no other way to truly appreciate it. I'm going to use it here in just a minute. I not only brought my computer; I brought Mr. Mike Juha, my programmer, and that really simplifies things. Mike is a senior in Dartmouth College, majoring in Engineering Science.

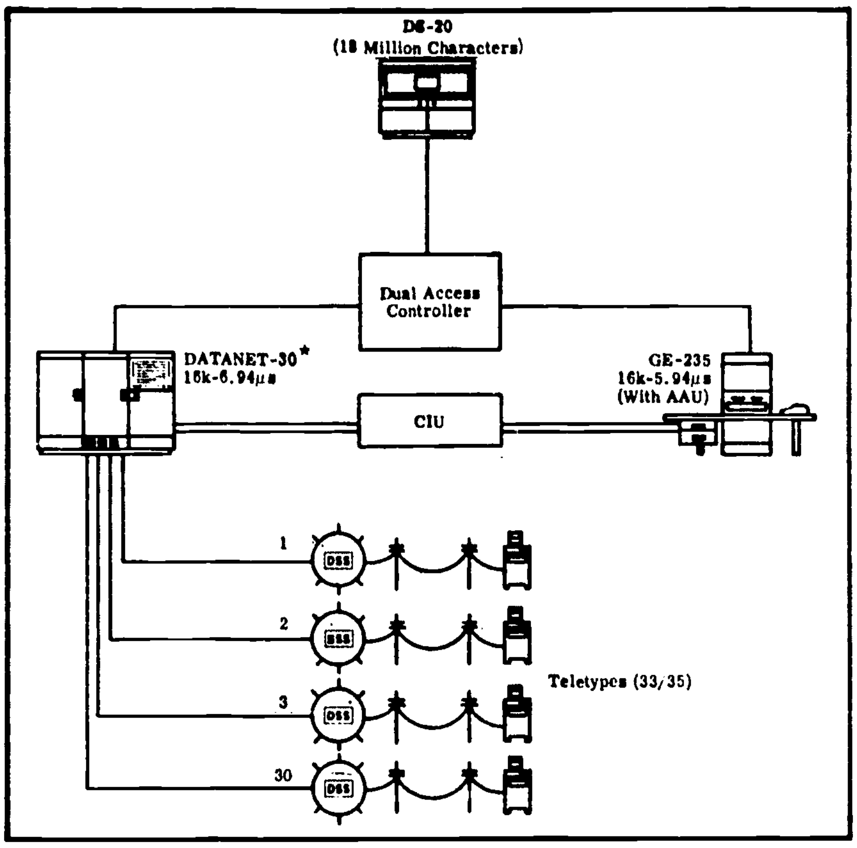


Figure 1. The Dartmouth-General Electric 265 Time-Sharing System.[2]

Our computer is a conversational computer so the first thing we'll do after calling on the telephone is to say "hello." The system replies, "Who are you?", i.e., what's your user number; what kind of programming language did you want to use? The conversation is opened. The system is so fast that I am not aware most of the time that there are other users.

At Dartmouth we have some 30 teletypes located around the campus. We have four in the engineering school which service 175 undergraduates, 60 graduate students and 25 staff members. Our basic philosophy is that Teletype time is "free," i.e., readily available for all to use. We seldom run into long queues to get on to the Teletypes. It is sometimes a little crowded in the afternoon during the week but computer use slacks off about 11:00 p. m. in any event.

[1] Kemeny, J. G. and Tribus, M., "The Dartmouth Time-Sharing System," *Journal of Engineering Education*, Vol. 56, No. 8, (1966) p. 326.

[2] G. E. Brochure CFB-440A.

* DATANET Reg. Trademark of the General Electric Co.

The system is active 20 hours a day, seven days a week. The 30 Teletypes on the campus service about 1,000 users who have approximately 5,000 programs in core. The system is memory bound. We have to "clean out" the 18-million character disc once a month. If you haven't used a program within the month it gets thrown away.

In a time-sharing environment if the Teletype doesn't talk to the user within 10 seconds (not 15 minutes or 3 days) he is unhappy.

AN EXAMPLE

Last February I needed an example problem for a workshop session of a course in digital simulation for practicing engineers in Sarnia, Ontario.[3] We were using the Dartmouth computer in Hanover, New Hampshire. A quick interrogation of my files produced a seven-stage extraction problem. Being pressed for time, I turned to Dan Frantz, a first-year graduate student. Dan was a physics major at Michigan. I described to him in 20 minutes how an extractor works. You know—solids go in and mix with water; the solution goes one way and the gang goes the other. Dan left me, went to his desk and wrote his computer-unit calculation in one hour. An hour and a half later he had programmed the unit calculation, got it working, immediately fitted it within an executive system for this kind of problem and in three hours, by the clock, he came back to me with the answers. His letter to me, the program and the answers are given in Appendix I. This illustrates very well the power of time-sharing in a programming environment.

As an engineer doing design I'm always generating alternatives but I want a quantitative evaluation of them. For example, what happens if I take 80 pounds per hour as the amount of water into the leaching system?

While this case is being run, I'd like to give you a little bit of the history of the development of time-sharing at Dartmouth and refer you to the April, 1966, issue of the *Journal of Engineering Education* which was published just in time to alleviate me from writing a whole lot of things down. In the lead article by Professors Donald Katz and Brice Carnahan, they view the future great impact of time-sharing computers in engineering education.[4] The last article by John Kemeny and Myron Tribus[5] describes this impact which is here today.

DEVELOPMENTS AT DARTMOUTH

One of the most fantastic success stories I know of in the development of the computer systems occured at Dartmouth under the direction of Professors John Kemeny and Thomas Kurtz. In March, 1964, the General Electric 235 computer hardware was unloaded from the truck and installed in the basement of College Hall. We had the usual trouble with air conditioning, etc. The equipment was in operation by April 1, 1964. On May 3, 1964, at 3:00 a. m., the time-sharing system operated with BASIC for 30 seconds! By June 1 of that year both the time-sharing system and the BASIC language compiler were operational in phase zero. In early June Professor Kemeny introduced BASIC and the system to the faculty of the college. The three summer months saw the establishment of the computer executive system at the Computation Center and both the time-sharing system and BASIC entered phase one of development. The system was then introduced to the entire college in September 1964 with some 20 Teletypes installed.

In Thayer School I presented two or three lectures on the system and BASIC, and staffed our computation room containing four Teletypes with undergraduate and graduate students to assist our students and staff in getting to know and use the new system. By February 1, 1965, I removed the students because they were no longer required! The system had become an integral part of our educational and research activities at Thayer School.

At Dartmouth, introductory programming, using the BASIC language, is taught to all of our freshmen in their mathematics course. About 90 per cent of our students (an all-male population) take a year of mathematics. I mentioned yesterday that this then led us to delete a course in numerical methods and computing from our engineering science core curriculum and put in a course in field theory, electricity and magnetism. But let's see what is really the impact on the students.

STUDENT IMPACT

In talking with Mike and some other students, they make statements such as, "We use the computer for anything that takes more than 30 minutes to calculate by hand; it's easier." Sophomore students in their first engineering course, ES 21, Introduction to Design, wrote and used a BASIC program which related torque, angular position, velocity and acceleration for the rotation of a human arm in raising itself and a weight in a vertical plane. They used these design calculations in their sophomore project work. We find that our students regularly use the computer to solve course problems without having staff members assign computer problems. The students are using the computer not because they

[3] Johnson, A. I., "Digital Simulation of Chemical Plants," *Chemistry in Canada*, April 1966, p. 16.

[4] Katz, D. L. and Carnahan, B., "The Place of Computers in Engineering Education," *Journal of Engineering Education*, Vol. 56, No. 8, (1966) p. 293.

[5] Kemeny and Tribus, op. cit., p. 326.

were told to do so, but simply because they see it as an easier way to do their work and they're always looking for an easier way to do things. They are developing an aptitude so that getting a numerical answer is trivial. The whole problem is: how to formulate problems, how to program them, how to set them up so that I know I will get answers. The students very rarely get into mathematical difficulties. When they do, they head for the staff member for help and they get them solved. We have mounted one Teletype on wheels and we use it in the classroom. In several courses that meet for a couple of hours, questions raised in the first hour have been programmed and completed in the fifteen-minute break between classes and the answers discussed in the second hour. Half of the students, in their courses in fluid mechanics and solid mechanics, write problems for data reduction and analysis of their laboratory work. We find a much closer tie between what is going on in the laboratory and the mathematical modeling because you can go back and forth so easily.

STAFF IMPACT

Now, what has happened to our staff? I'm not so much impressed when Paul Shannon uses the computer as I am when some of our senior staff members who have never used a computer before do so. For example, a senior professor in civil engineering, whose field is water and waste treatment, has taught himself BASIC and discovered that in 15 minutes he can do simple linear regressions that used to take him three to five days; or three of our staff, having no previous computer experience, who got together and offered a computing course for Dartmouth alumni in the construction field. It is this impact that impresses me. I did a quick tabulation of our staff members and found that a little over 50 per cent of our staff members use the computer. Within that 50 per cent there are five that use it very heavily and within that five there is one who takes 90 per cent of the machine time!

Time-sharing man-machine interaction produces a situation in which staff and students can implement their concepts, program their mathematical model, debug the programs and obtain answers 10 to 100 times faster than a person operating in an efficient batch-processing system. These figures are not just based on looking at Dartmouth but also on some surveys that were done in industry. The industrial surveys indicated that it was 40 to 100 times faster by the clock between the statement of a problem and the answers compared to a conventional batch system.

A computer is now as close as your telephone. I have one here, thanks to the work of a lot of people. I helped to teach a course in digital simulation to a group of practicing engineers in Sarnia, Ontario this fall and we ran on the Dartmouth computer as part of the course work.[6] I am teaching a course in another industrial concern and will install my computer terminal and operate. In the chemical industry you have some companies right now who have extensive internal time-sharing capabilities. Tremendous strides are being made in the automatic analysis of data from analytical expirements. If you look for repetition, chemical analysis is a fertile field. So now we see automatic programming of experimental equipment such as an electron probe microscope taking this information immediately into the core of the computer via telephone lines and processing it. I was at one research center where a man made the statement that he can now generate in one week all the experimental data ordinarily found in a Ph. D. thesis!

COSTS

Time-shared means cost-shared and in calculating the cost of a time-sharing system you have these three basic elements: the cost of the remote terminal, the cost of the telephone link, and the cost of core time on your big computer. This opens up a tremendously increased number of possible users. However, while the rates go lower and lower, the total price tag goes higher and higher. Two years ago Dartmouth had an LGP 30 and an IBM 1620 computer which cost us roughly $1,100 a month. Now we have 30 Teletypes on campus, each of which costs $130 a month for a total of $3,900 a month just for our remote terminals. And we're contemplating a next generation of computers in which the maintenance cost is about $100,000 a year. Perhaps these few figures will highlight the very serious questions about who is going to pay for this tremendously increased functional capability that is being implemented in our colleges and engineering schools.

CONCLUDING REMARKS

Well, I want to finish up very briefly by indicating one type of design effort which is possible today because of computers and one which I think is an indication of what is going to happen tomorrow.

Last year I had the real pleasure of working with the five members of the Chemical Engineering department at McMaster University and their eight senior chemical engineering students. Working closely with people from Canadian Industries Limited, we set ourselves the task of mathematically modeling C. I. L.'s 300 ton/day sulfuric acid plant located in Hamilton, Ontario. We started in December, got going by the middle of February and com-

[6] Johnson, op. cit., p. 16.

pleted the study on April 6.[7] The resulting PACER simulation represented the simultaneous solution of approximately 500 equations, many of them highly non-linear, given about 200 constants and solving for about 1,000 unknowns.

I will simply say that we did it and one of the things that gave me the most pleasure was talking to a student just before final presentation who said, "When I first started this project I thought it was impossible. I no longer do. It's just difficult."

In my graduate level design courses I am now asking my students to realistically model an entire desalinization plant within three to four weeks. This is only possible in a time-sharing enivronment. Time-sharing is having a tremendous impact and I think that as we go into the next five years we will develop more users and terminals which have much more man-machine interaction, and thus evolve an almost completely different mode of computer utilization in engineering education and design.

BRAUN: *We've had two terminals into the GE Datanet system for about five months. I'm not a chemical engineer and I don't really know what chemical engineers do, but one of our senior and most conservative chemical engineering professors has brought it into his unit operations lab. As I understand it, the students used to make some measurements and sit down at a table in the lab with their slide rules for an hour or two, make some calculations, and then go back. Now, every Wednesday and Friday afternoon the professor runs down and puts a cart underneath our Teletype terminal and runs it into the lab, and the students now make some measurements, run a couple of programs they have developed and in a matter of minutes go back and modify their measurements. He is completely sold on this thing; I think probably if we got rid of these machines, he and a number of the other students would pull the walls of the place down!*

I would like to take some slight exception to a comment made about the price of this. These two terminals were costing us $5,000 a month because the machines were up almost twenty-four hours a day, six days a week. This is pretty nearly half of what it cost us to run our 7040, and that's only two terminals running almost twenty-four hours a day. We now have a lock on the telephone dial. The lock distresses me, because the students are not using it nearly as effectively as the people at Dartmouth. We've managed with the locks to get the cost down to about $2,200 a month. That's the problem. I think that this is the wave of the future, but somebody's got to give some serious thought to ways of cutting the price down by orders of magnitude.

PAYNTER: *Two questions, one serious, one humorous. The serious one—are you people thinking at all of really small, portable consoles that students could actually carry with them?*

SHANNON: *No. We are looking very much at the design of 1,000-bit per second remote terminals. Now the question is: can we build a box that sells for under $500 which will put out plus and minus 10-volt analogs, plus and minus 10-volt electrical signal, so that I can plot on a standard oscilloscope. We may have to buy a memory scope initially, so that we can drive a standard analog plotter, but the feeling is that if the engineer can get an answer out to 10 per cent on an x-y plot for a couple of variables, he won't want to see 90 per cent of the stuff he now prints out. And so it's the development of this next generation of program-controlled hardware that we're looking at.*

PAYNTER: *The humorous one was, has anybody at Dartmouth suggested using terminals to test and teach, thereby putting the faculty either on sabbatical or emeritus?*

SHANNON: *Well, not quite, because there's an awful lot involved here. I find myself using the following words trying to get across some of the ideas. The words are, "mathematical hardware." When Dean Tribus asks me what we do, I say We build mathematical hardware. And computer programming is the development of mathematical hardware. And as we are able to define functionally the components of our hardware and build up the unit calculations and have standard routines, we can then go to the building of automatic machinery. Now automatic machinery is modular, functionally designed, and highly specific. So I think there's quite a good analog here between the generation of problem-oriented languages now under development and the sort of automatic machines we see sitting around in plants. Machines also have capital costs, design ideas, maintenance costs, operating costs, all of these things which are all true of computing work, and this is one way I try to get across, not to an audience like this, but to those who have no feel at all for computers, what is actually going on. The whole PACER system might be said to represent a format for mechanically implementing the standard mathematical techniques known today.*

QUESTION: *You mentioned somebody who is generating data in one week that originally took about*

[7] Shannon, P. T., et. al., "Computing Simulation of a Sulfuric Acid Plant," *C. E. P.*, Vol. 62, No. 6 (1966) p. 49.

a year without the computer. I think I talked to the same man, and he also mentioned that, being a Ph.D. candidate, his school refuses to accept this work for dissertation, or at least is very reluctant to do so. So my question is directed to you and all the other educators, is this going to be a problem?

SHANNON: *My answer to that is yes. Particularly, one has to come to the realization that a legitimate product for an engineer today is information. In fact, I think a tremendous percentage of our engineers are really technical information handlers, basically, in their job function. But if you say to a man who thinks only in terms of physical embodiment of his concepts, I've got a student who has written this fantastic computer program, and he says, "Yes, but he's done no design," I say, what do you mean he has done no design? Well, he didn't conceive of anything and try it and modify it, generate alternative ways to do something and all the rest of this. We heard so much of this yesterday but, in fact, this does involve exactly the design function, the design process. Creating these programming languages, creating these mathematical systems involves the same set of ideas and principles and techniques and problems, but it is not recognized as such as yet. And this is another reason I talk to my engineering educator friends about mathematical hardware, to try to get them into this frame of mind. But it is a problem and it is not recognized. Another thing that is not recognized is the fact that computer programs should be treated exactly the same way as any other scientific publication. The first thing on it is the man's name, referenced, documented, and you don't pick up anybody else's work and use it without giving credit. And this has got to go forward; it's moving much more in this direction now.*

QUESTION: *I'd like to draw a distinction between generating ideas and translating them into computer languages. The man who generates the ideas deserves the credit; the man who translates them into computer languages is merely performing the tasks that can be done in a pedestrian way.*

SHANNON: *That's right. You don't want to get involved in coding. There's a great deal of difference between coding, writing out the detailed procedural instructions, and the process of trying to generate out the algorithm, the functional form of your system. How it's all going to be put together, how to analyze it and study the information flow, all of these things are not coding problems.*

QUESTION: *I'd like you to extend that to one more question, please, because I think there's an important factor left out in this time-sharing discussion. And I think the speaker should say something about the considerations of the incorporation of display scopes in the general time-sharing system.*

SHANNON: *I visualize the next couple of generations of time-sharing equipment as going through several orders of magnitude as far as the remote stations. At level one, we had the DAC-I, "sketchpad" kind of console with the light pen, a million dollars of hardware on the end of your remote station. For every one of these systems I visualize ten small computers, actually complete remote computers with very fast transmissions into larger processors. And for every one DAC and ten small computers, a hundred systems which will have very limited facilities, say three-inch scope, able to print at 100 lines a minute. And then we are going to see a thousand Teletype-like terminals. So, we are going to get this triangle with decreasing costs and more users at each level.*

APPENDIX

FEBRUARY 23, 1966

TØ: PAUL T. SHANNØN
FRØM: DANIEL R. FRANTZ
RE: SEVEN STAGE LEACHER USING 'PACER'

ENCLØSED ARE MATERIALS CØNCERNING THE SEVEN STAGE LEACHER THAT HAS BEEN ADDED TØ THE 'PACER TIME-SHARING MANUAL':

1. STATEMENT ØF PRØBLEM
2. INFØRMATIØN FLØW CHART
3. LISTING ØF UNIT CALCULATIØN
4. SINGLE STAGE EXTRACTIØN
   A. PRØGRAM AND DATA
   B. RESULTS
5. SEVEN STAGE LEACHER
   A. PRØGRAM AND DATA--INCLUDING AUXILIARY SUBRØUTINE 'SETH2Ø' LISTING
   B. RESULTS

NØTE THAT SUBRØUTINE 'SETH2Ø WAS WRITTEN TØ SPEED CØNVERGENCE BY FIRST FILLING THE EXTRACTIØN CELLS WITH WATER AS THEY INDEED WØULD BE IN ACTUAL ØPERATIØN. THE SAME EFFECT CØULD HAVE BEEN PRØVIDED BY ADDING THE WATER TØ THE DATA, BUT THIS APPRØACH SEEMS SØMEWHAT MØRE FLEXIBLE.

THE RESULTS ØF THIS SIMULATIØN ARE IN EXACT ACCØRD WITH THE ANALYTICAL RESULTS FØR THIS LINEAR CASE. IT IS NØTEWØRTHY, HØWEVER, THAT WHILE ANALYTICAL SØLUTIØNS TØ SUCH A PRØBLEM DØ NØT EVEN EXIST FØR NØN-LINEAR CASES, ALL THAT IS REQUIRED TØ MØDIFY THE 'PACER' SUBRØUTINE FØR NØN-LINEARITY IS THE REDEFINITIØN ØF ØNE EQUATIØN. (REFERENCE FØR ANALYTICAL SØLUTIØN: CHAPTER XI, "ELEMENTS ØF CHEMICAL ENGINEERING", W.L. BADGER AND W.L. MC CABE.)

ANØTHER PØINT ØF INTEREST IS THE AMØUNT ØF TIME REQUIRED TØ DEVELØP THIS SUBRØUTINE. THE SELECTIØN AND DISCUSSIØN ØF THE PRØBLEM WITH A NØN-CHEMICAL ENGINEERING STUDENT TØØK ØNE HALF HØUR; CØDING INTØ AN 'ALGØL', 'PACER' TYPE UNIT CALCULATIØN ØCCUPIED ØNE AND A HALF HØURS; TYPING AND DEBUGGING AT THE TELETYPE REQUIRED ANØTHER HØUR AND HALF--A TØTAL ØF THREE AND A HALF HØURS FØR A STUDENT WHØ HAD NEVER HEARD ØF A LEACHER BEFØRE.

LEACHING

AN EXTRACTIØN BATTERY CØNSISTS ØF 7 DØRR THICKENERS. THERE IS FED INTØ THE NØ. 7 THICKENER (STRØNG SØLUTIØN END) 20 TØNS PER HØUR ØF MATERIAL TØ BE LEACHED, ANALYZING:

80 PERCENT BY WEIGHT INSØLUBLE GANGUE
9 PERCENT HIGHLY SØLUBLE SALT (A 28 PERCENT SØLUTIØN IS SATURATED AT THE TEMPERATURE ØF LEACHING)
11 PERCENT WATER

THERE IS PUT INTØ THE NØ. 4 THICKENER PER HØUR 15 TØNS WEAK MØTHER LIQUØR ANALYZING:

3 PERCENT SALT
97 PERCENT WATER

THERE IS PUT INTØ THE NØ. 1 THICKENER PER HØUR 40 TØNS ØF WATER.

EQUILIBRIUM BETWEEN UNDERFLØW AND ØVERFLØW MAY BE ASSUMED TØ BE 2 TØNS ØF WATER (EXCLUSIVE ØF DISSØLVED SALT) PER TØN ØF INSØLUBLE GANGUE. LIQUID AND GANGUE MØVE THRØUGH THE BATTERY IN CØNTINUØUS CØUNTERCURRENT FLØW.

CALCULATE

A) QUANTITY ØF STRØNG SØLUTIØN PRØDUCED PER HØUR.
B) CØNCENTRATIØN ØF STRØNG SØLUTIØN.
C) WEIGHT PER HØUR ØF SØLUBLE SALT DISCHARGED FRØM NØ.1 ALØNG WITH INSØLUBLE GANGUE.

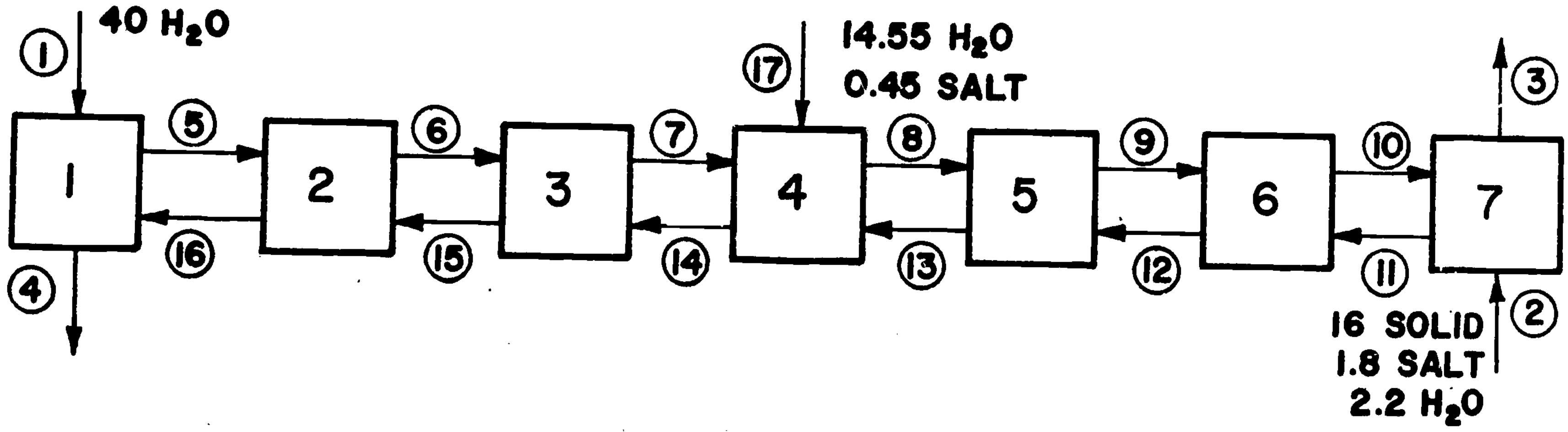


FIGURE 2   SEVEN STAGE LEACHER

```
EXTR      14:51   JUNE 17,1966

    PRØCEDURE EXTRAC;  BEGIN REAL FR,RAT,WAT,SLT,SØL,A,B;  INTEGER I,J;

    CØMMENT THIS PRØCEDURE SIMULATES AN EXTRACTIØN CELL.
            IT MIXES ALL THE INPUTS AND DISTRIBUTES THE RESULTANT
            TØ THE ØUTPUT STREAMS AS FØLLØWS:
            FIRST ØUTPUT:     ØVERFLØW STREAM, RECEIVES THE EXTRA WATER
                              AND SALT, IF ANY.
            SECØND ØUTPUT:    THE CØNCENTRATED ØUTPUT, TAKES A PØRTIØN ØF
                              THE SØLID (AS DEFINED BY THE EQUIPMENT
                              PARAMETERS BELØW), SØME WATER AND SALT.
            THIRD ØUTPUT:     (IF PRESENT) RECEIVES THE REMAINING SØLID,
                              AND A PRØPØRTIØNAL AMØUNT ØF WATER AND SALT

            IF THERE IS NØT ENØUGH WATER TØ MEET THE REQUIREMENTS FØR
            ØUTPUTS 2 AND 3, THEY WILL BE PRØ-RATED ACCØRDING TØ THE
            THE AMØUNTS ØF SØLID EACH IS TØ RECEIVE.

            STREAM FØRMAT                 EQUIPMENT FØRMAT
            1. STREAM NUMBER              1. EQUIPMENT NUMBER
            2. AMT. SØLID                 2. FRACTIØN SØLID TØ ØUTPUT 2
            3. AMT. WATER                 3. RATIØ ØF WATER TØ SØLID
            4. AMT. SALT
            5. FRACTIØN SALT SØLUTIØN;

     SLT:=WAT:=SØL:=0;
     CØMMENT TRANSFER TØ PRØGRAM VARIABLES AND FIND
             TØTAL SØLID, WATER AND SALT INCØMING;
     RAT:=EN[NE,3];
     FR:=IF NØUT=3 THEN EN[NE,2] ELSE 1;
     FØR I:=1 STEP 1 UNTIL NIN DØ BEGIN
       SØL:=SØL+STRMI[I,2];
       WAT:=WAT+STRMI[I,3];
       SLT:=SLT+STRMI[I,4];  END;

     STRMØ[2,2]:=A:=FR*SØL;
     STRMØ[3,2]:=B:=(1-FR)*SØL;
     IF RAT*SØL<=WAT THEN BEGIN
       CØMMENT DESIRED RATIØ CAN BE SUPPLIED, DØ SØ;
       STRMØ[2,3]:=A:=A*RAT;
       STRMØ[2,4]:=A*SLT/WAT;
       STRMØ[3,3]:=B:=B*RAT;
       STRMØ[3,4]:=B*SLT/WAT;   END
     ELSE  BEGIN
       CØMMENT NØT ENØUGH WATER, PRØ-RATE IT AND THE SALT;
       STRMØ[2,3]:=FR*WAT;
       STRMØ[2,4]:=FR*SLT;
       STRMØ[3,3]:=(1-FR)*WAT;
       STRMØ[3,4]:=(1-FR)*SLT;   END;

     CØMMENT FIRST ØUTPUT IS ØVERFLØW;
     STRMØ[1,2]:=0;
     STRMØ[1,3]:=WAT-STRMØ[2,3]-STRMØ[3,3];
     STRMØ[1,4]:=SLT-STRMØ[2,4]-STRMØ[3,4];

     CØMMENT NØW CALCULATE SALT CØNCENTRATIØN FØR ALL STREAMS;
     FØR I:=1 STEP 1 UNTIL NØUT DØ  STRMØ[I,5]:=
       IF STRMØ[I,4]=0 THEN 0 ELSE STRMØ[I,4]/(STRMØ(I,4]+STRMØ[I,3]);
    END ØF EXTRAC;
```

```
SETH2Ø    14:55   JUNE 17,1966


    PRØCEDURE SETH2Ø;  BEGIN INTEGER I,J,N;  REAL A;

    CØMMENT THIS SUBRØUTINE IS USED TØ INITIALIZE THE WATER
            CØNTENT ØF STREAMS IN A SERIES ØF EXTRACTIØN CELLS.

            IT TAKES THE WATER THAT IS CØNTAINED IN THE INPUT STREAM
            AND FEEDS IT TØ ØTHER  STREAMS, AS DESCRIBED BELØW.

            THE STREAMS TØ BE " WATERED" ARE GIVEN IN THE SPECIAL
            DATA BLØCK "EWAT" IN THE FØRMAT:
               DATA EWAT:=N,S1,S2,...,SN;
            CØMMENT WHERE N = NUMBER ØF STREAM NUMBERS FØLLØWING
                         SJ = THE NUMBERS ØF THE STREAMS TØ BE WATERED

             THE STREAM FØRMAT ASSUMED IS:
              1. STREAM NUMBER
              2. AMT. SØLID
              3. AMT. WATER
              4. AMT. SALT
              5. FRACTIØN SALT SØLUTIØN;

     CØMMENT A IS AMØUNT ØF WATER TØ FEED;
     A:=STRMI[I,3];

     CØMMENT RESTØRE DATA BLØCK IN CASE THIS EQUIPMENT IS USED
             MØRE THAN ØNCE;
     RESTØRE(EWAT);
     READATA(EWAT,N);
     FØR  I:=1 STEP 1 UNTIL N DØ BEGIN
       READATA(EWAT,J);
       SN[J ,3]:=A;
       END;

    END ØF SETH2Ø;
```

```
LIST


LEACHR    17:40   JUNE 9,1966

100 PRØCEDURE EXTRAC;BEGIN REAL FR,RAT,WAT,SLT,SØL,A,B;INTEGER I,J;
110 SLT:=WAT:=SØL:=0;RAT:=EN[NE,3];FR:=IF NØUT=3 THEN EN[NE,2] ELSE 1;
120 FØR I:=1 STEP 1 UNTIL NIN DØ BEGIN SØL:=SØL+STRMI[I,2];WAT:=WAT+
130 STRMI[I,3];SLT:=SLT+STRMI[I,4];END;STRMØ[2,2]:=A:=FR*SØL;STRMØ[3,2]:=
 140 B:=(1-FR)*SØL;IF RAT*SØL<=WAT THEN BEGIN STRMØ[2,3]:=A:=A*RAT;
150 STRMØ[2,4]:=A*SLT/WAT;STRMØ[3,3]:=B:=B*RAT;STRMØ[3,4]:=B*SLT/WAT;
 160 END ELSE BEGIN STRMØ[2,3]:=FR*WAT;STRMØ[2,4]:=FR*
170 SLT;STRMØ[3,3]:=(1-FR)*WAT;STRMØ[3,4]:=(1-FR)*SLT;END;STRMØ[1,2]:=0;
 180 STRMØ[1,3]:=WAT-STRMØ[2,3]-STRMØ[3,3];STRMØ[1,4]:=SLT-STRMØ[2,4]-
 190 STRMØ[3,4];FØR I:=1 STEP 1 UNTIL NØUT DØ STRMØ[I,5]:=IF STRMØ[I,4]=0
 200 THEN 0 ELSE STRMØ[I,4]/(STRMØ[I,3]+STRMØ[I,4]);END;
300 PRØCEDURE SETH2Ø;BEGIN INTEGER I,J,N;REAL A;A:=STRMI[1,3];
310 RESTØRE(EWAT);READATA(EWAT,N);FØR I:=1 STEP 1 UNTIL N DØ BEGIN
320 READATA(EWAT,J);SN[J,3]:=A END;END;
400 PRØCEDURE EQCAL;IF NECALL[NE]=3 THEN EXTRAC ELSE SETH2Ø;


ØLD
ØLD PRØBLEM NAME--D1LECH***

READY.

LIST


D1LECH    17:42   ØJUNE 9,1966

410 PRØCEDURE PRUN;IF NECALL[NE]=3 THEN PRINT("","EXTRAC","") ELSE
420  PRINT("","SETH2Ø","");
1000 CØMMENT SEVEN STAGE LEACHER;
1010 DATA D1:=21,50,1;
1020 DATA D2:=8,6,3,17,5;
1030 DATA D3:=8,
1031 1,3,1,16,-5,-4,0,
1032 2,3,5,15,-6,-16,0,
1033 3,3,6,14,-7,-15,0,
1034 4,3,7,13,17,-8,-14,
1035 5,3,8,12,-9,-13,0,
1036 6,3,9,11,-10,-12,0,
1037 7,3,10,2,-3,-11,0,
1038 8,4,1,-1,0,0,0;
1040 DATA D4:=8,
1041 1,1,2, 2,1,2, 3,1,2, 4,1,2, 5,1,2, 6,1,2, 7,1,2, 8,0,0;
1050 DATA D5:=3,
1051 1,0,40,0,0,
1052 2,16,2.2,1.8,.45,
1053 17,0,14.55,.45,.03;
1060 DATA D6:=$-4,$-4,$-4,$-4,$-4;
1070 DATA D7:=8, 8,1, 7,2, 6,2, 5,2, 4,2, 3,2, 2,2, 1,3;
1080 DATA EWAT:=12,5,6,7,8,9,10,16,15,14,13,12,11;
1090 END;
```

```
EDIT WEAVE PACER-***, LEACHR***, DILECH***

READY.


RUN



DILECH     17:47

DARTMØUTH ALGØL.



DATA FØR RUN NUMBER 21

PRØCESS MATRIX

 1     EXTRAC    1      16     -5     -4      0
 2     EXTRAC    5      15     -6     -16     0
 3     EXTRAC    6      14     -7     -15     0
 4     EXTRAC    7      13      17    -8     -14
 5     EXTRAC    8      12     -9     -13     0
 6     EXTRAC    9      11     -10    -12     0
 7     EXTRAC    10     2      -3     -11     0
 8     SETH2Ø    1     -1       0      0      0

CALCULATIØN ØRDER
EQUIP  MØDE
 8      1
 7      2
 6      2
 5      2
 4      2
 3      2
 2      2
 1      3

EQUIPMENT PARAMETERS

 1             1             2
 2             1             2
 3             1             2
 4             1             2
 5             1             2
 6             1             2
 7             1             2
 8             0             0
```

```
STREAM VARIABLES

 1             0              40              0              0
 2             16             2.2             1.8            .45
 3             0              0               0              0
 4             0              0               0              0
 5             0              0               0              0
 6             0              0               0              0
 7             0              0               0              0
 8             0              0               0              0
 9             0              0               0              0
 10            0              0               0              0
 11            0              0               0              0
 12            0              0               0              0
 13            0              0               0              0
 14            0              0               0              0
 15            0              0               0              0
 16            0              0               0              0
 17            0              14.55           .45            .03

CØNVERGENCE CRITERIA
 .0001         .0001          .0001           .0001          .0001

CØNVERGENCE IN 39   LØØPS


RESULTS FØR RUN NUMBER 21

STREAM VARIABLES

 1             0              40              0              0
 2             16             2.2             1.8            .45
  3             0              24.75           2.0962         7.80819 $-2
  4             16             32              .153149        4.76312 $-3
  5             0              40              .191436        4.76312 $-3
  6             0              40              .430732        1.06536 $-2
  7             0              40              .729875        1.79199 $-2
  8             0              54.55           1.50538        2.68553 $-2
  9             0              54.55           2.06038        3.63958 $-2
  10            0              54.55           3.00668        5.22385 $-2
  11            16             32              2.71024        7.80819 $-2
  12            16             32              1.76377        5.22385 $-2
  13            16             32              1.20866        3.63958 $-2
  14            16             32              .883084        2.68553 $-2
  15            16             32              .5839          1.79199 $-2
  16            16             32              .344586        1.06536 $-2
  17            0              14.55           .45            .03

CØNVERGENCE CRITERIA
 .0001         .0001          .0001           .0001          .0001


TIME = 37 SECS.
```

THE NEED FOR COMPUTERS IN TEACHING AND RESEARCH

H. M. PAYNTER
Massachusetts Institute of Technology
Cambridge, Mass.

INTRODUCTION

As high-speed computers seem here to stay, albeit becoming ever larger, faster, more hybridized and interactive, it now seems useful and necessary to attempt to assess their place in society. If they are, indeed, as revolutionary an innovation as gunpowder, movable type, the steam engine, etc., then in particular we should try to forecast their impact upon our engineering schools.

Now I happen to belong to that strange cult which conceives the design mystique as the natural highest calling of the engineer, and, in truth, of all professionals and artists. So I cannot view the teaching of design as somehow distinct from teaching, research, and scholarship in general. And for me, at least, computers in various guises are as simple necessities as pencils for the poet and scalpels for the surgeon.

To avoid some of the stigma of presumptuous pretense, and also simply to have fun, I have chosen to cast my pearls in the form of a fictitious dialog. Perhaps thereby I can express with impunity some strong feelings on the evolving role of computers-on-campus.

Prologue and Cast of Characters

The following imagined conversation takes place between two protagonists:

> *ARDENT PROPONENT:* a young man of the new breed, born to a world in which television, space-flight and large computers seem as natural as air and water; and
>
> *SKEPTICAL CONSERVATIVE:* an older man, mentor of the foregoing, alert to the exploding technology and the main thrust of the systems approach, but whose natural milieu is philosophy and scholarship.

Henceforth they are called simply PRO and CON, respectively. Together with some of their school colleagues, they are responsible for undergraduate and graduate curriculum development and other teaching and research activities in the areas of system dynamics and control, and the applications of computers to engineering analysis and design. Both have spent countless hours over the last decades in interdisciplinary workshops and on faculty committees devoted to such subjects. PRO is Director of the Computer Center; CON is head of the newly organized Department of Electromechanical Systems Engineering. PRO has just returned to XY Tech from the Conference on the Impact of Computers on Education in Engineering Design.

The scene opens:

PRO: Well, thanks for sending me to the conference. At least this one was concerned with new uses for computers and not just new models, programs and languages.

CON: You don't have to go to Chicago to learn that, I hope. Did anybody say anything significantly different or differently?

PRO: At least a theme was well set at the start. Dean Everitt charged the conference with the professional obligation to crystallize the fundamental concepts of computer-aided design. Licklider and Miller said we must learn how to teach students *now* to fully exploit the computers of decades hence, which will be unlike anything now existing. Licklider went on to say these machines undoubtly will deal with more than numbers and alphanumerics, and will rather be true information networks. Chuck Miller described the newer new-man of 1976: a master designer more akin to the architectural master planner-builder than the design engineer of today.

CON: Wait a moment! Surely these ideas didn't all go unchallenged. I have heard Licklider wax poetic over the on-line intellectual community—a more attractive wording than Martin Greenberger's information utility. But aren't there a few steps along the road which are not even now equipment-limited? You know that I think the book was the greatest teaching machine ever invented! Maybe Lick thinks people have forgotten how to read as they have forgotten how to spell?

PRO: That's not the point! Lick, and others, too, made much of interactive graphic displays used for self-communication and for inter-communication. I don't see how any ordinary sort of book can fulfill that role. And you, yourself, know what we are doing with computer-aided teaching of dynamic system behavior. Incidentally, much talk went on regarding on-line dynamic modeling and system simulation. This was put in contrast to the conventional, more static, view of engineering.

CON: But a passive attitude and a static view have never been natural either to children, or to Americans, for that matter! Change and dynamism are not exactly new. What may possibly be new is the ability to describe and to control change; to appreciate and to model dynamic situations amidst complexity and confusion. Certainly for us who were teethed on analog computers and even on thinking by analogy, on-line dynamic modeling and real-time simulation are not new experiences. We have attempted to see behavior before building for some time.

And another point before castigating statics: remember that dynamics in 3-space becomes statics in 4-space; the Hamilton-Lagrange dynamics of n-degree-of-freedom systems becomes statics when time is adjoined as a state-variable.

PRO: Of course, I generally agree with you. You must think that most of such people are mere Johnny-come-latelies who only associate Lagrange with multipliers and Hamilton with quadternions. But another important point was raised: that there is an analogy between the man-computer and the heuristic-algorithm situation. In particular, Licklider set design in the context of problem-solving and in turn emphasized that problem-solving ultimately demands successful formulation: the asking of the right questions.

CON: Whoa there! Don't credit the problem-solving or heuristic approach to Lick or even to Marvin Minsky, for that matter. I seem to recall that it was the mathematician George Polya, present elder statesman of analysis, who made the word "heuristic" fashionable. If the computer-bugs find his ideas useful, so much the better for them!

But his reminds us of our recurrent fear: the possible adverse effect of computers on analytical skills. Aren't we always likely to fall into the trap: "Why think, if we can compute?"

PRO: But you know yourself that computers are no more at fault on this count than logarithms or nomograms. There is no apparent reason why the mere existence of one tool should necessarily stultify another. Moreover, any artful use of computers should enhance the analytical gifts of the user.

CON: Well, I agree with you in part. But I'll use your point to make another. You talk as if everyone at the conference thought always of computers as large, general-purpose, digital computers. . .

PRO: . . . except for Jim Reswick. . .

CON: And also, I would suppose, Hank Paynter. But, to go on, surely among the most remarkable developments of the last few decades has been the incorporation of computers, large and small, analog, digital, hybrid, directly into prototype systems, frequently as special-purpose devices. For the life of me, in this connection, I find it hard to distinguish between computers, proper, and such elements as instruments, signal conditioners and controllers. Certainly all must play a role in design.

PRO: As I said, Jim Reswick in particular made this point at the conference. But I would like to get across another recurrent point: nearly everyone viewed present and future computers as exemplary means to enhance the creative talents and decision-making skills of the engineer, both in school and in industry. Through computer simulation the design engineer (or engineer-in-training) could rapidly acquire these skills, explore a wider range of alternatives, and substantially optimize a tentative or final design.

CON: I am always bothered by this word "creative" which keeps cropping up at all meetings on design. If we are being honest about design as it has actually been practiced, why attach such a premium on creativity? Or, similarly, on novelty in research? Shouldn't the goal of honest research effort be better understanding, at least for the researcher, if not for the world?

Are not then old things and old ideas valuable? Just because the simple lever was invented (or better, just thoroughly understood) by Archimedes more than 2,000 years ago makes it no less valuable or so much easier to improve upon (perhaps we could make it 200% efficient!). My guess is that when Licklider's on-line community gets going there will justly be a number of voices from the grave!

Aren't all you people pussy-footing around the legitimate meaning of creative engineering and design? Wouldn't it be more accurate to say that a prime task of the engineer is to produce (or at least help to produce) acceptable, efficient, and competitive designs, whether aided by computer or not. Novelty enters the picture only because the task changes and new needs and possibilities present themselves. But no single engineer should take unto himself too much credit for that!

It seems to me that the distinction which some people make between ordinary engineering and creative engineering is that which I would make between bad engineering and engineering.

And another point, if you don't mind my shouting you down. You let slip with that nasty word "optimization." Shouldn't the rule of parsimony

—Ockham's razor—also apply occasionally? Why shouldn't we search more often for minimal models: models which lead us to the correct decisions most efficiently, with the least (and not the most!) degree of elaboration. Surely, if I granted you a sum of money, you wouldn't instinctively want to spend it as quickly as possible?

PRO: I'm not so sure about that! If I were too miserly you would cut my budget next year! Which reminds me: it was the question of budgeting for computer needs which sent me off to the conference. It strikes me that nobody attempted to come to grips with the question of computer costs in academic budgets for the future. The speakers and audience seemed still basking in a honeymoon glow. Nobody seems to have mentioned figures like the ones we have come up with: $100 to $400 per year of computer cost for every Tom, Dick and Harry at XY Tech.

CON: Before we get back to that item, I would like to summarize our thoughts before you left for the meeting and then ask you if you have changed your opinion regarding any of these.

We had generally agreed that the primary responsibilities of the school lie in the areas of teaching and research. Relative to engineering design, this implies that we may consider the teaching of design and the design of teaching methods, likewise research in design and design of research procedures and experiments.

As to computers we felt that, directly related to computer-aided design, are computer-aided teaching, and computer-aided research, including experiments in fully automated experimentation: machines to produce fundamental scientific knowledge without direct human intervention!

What are the objectives in all this? Simply the goals of every university: the education of students, researchers, and faculty. Particularly we felt it important to stress that a school must help each student, as soon as he shows up at the door, to cultivate two attitudes (with or without the assistance of a computer!):

1. A strong, self-confident, personal style and deliberative purposeful response: in all, a sense of unity and integrity; and
2. A lasting awareness and speculative curiosity: in sum, a life-long zest for learning and accomplishment.

I think both of us felt that, should computers ever interfere with these objectives, then you and I, at least, must set about to develop something better and more wholesome which can ultimately supplant them. What do you say now to these lofty ideals?

PRO: Nothing I heard or saw would cause me to declare war yet! Both industrial and academic people foresee benefits far outweighing liabilities. I'd like to summarize some thoughts of mine which were certainly sharpened by this meeting.

(1) In the absence of clear understanding of collective, organic design procedures, the experiments in design on campuses yield at least as significant information as industrial counterparts. Typical school design projects are smaller, directly involve far fewer people, and are much more closely monitored (contrary to belief in some quarters).

(2) Computers have been used on campuses for formal scientific purposes far longer than in industry. By natural evolution, then, computer-aided, rationally analyzed student designs have become more commonplace than is generally realized.

(3) Schools have, generally speaking, done much better than industry in maintaining a balance between analog and digital hardware and utilization. But, with growing appreciation of the singular benefits of hybrid computation, a new rebirth of significant school-based hardware activity is a likely event.

(4) Schools are woefully remiss in laboratory instruments and control relative to counterpart industry. But campuses will soon see the start of revolutionary developments in on-line computer control of research experiments.

(5) Clear patterns have been established for successful on-line, computer-aided teaching and the design and use of upgraded, open-ended teaching machines. Only the overwhelming costs of utilization, followed by more modest requirements of faculty indoctrination, presently limit the impact of such developments. Remove these barriers and some of Licklider's most colorful predictions may yet become understatements.

(6) Finally, via computers, several aspects of the real engineering world may be brought into the classroom, exposing students to situations which would otherwise be impossible to experience. In particular, reasonable, flexible, articulate models of large and sophisticated systems (for transportation, power, etc.) can be made readily available.

CON: I could comment endlessly on this but we both have classes coming up, so let's get back to work!

FENVES: *One of your points was that there is a kind of osmosis from research use of computers to teaching use and computer-aided design teaching. Are you sure that there is such a thing, or is* *this just wishful thinking? My experience has been that there is something the psychologists call a "logic block." We get bright young Ph.D.'s who have used the computer for their research work and who continue using the computer exclusively for the analytical research work, but the thought of using the computer in classrooms just doesn't seem to occur to them. Would you like to comment?*

PAYNTER: *We really have nothing to do with that process once you put the tools in their hands. Once every freshman at a university—which is a liberal arts school—has access to computing methods, I think you can leave it safely up to him to exploit these rather fully. There's another aspect of the research, and that is the really close interplay between the research-experimentation, instrumentation-type use of computers and the direct use in teaching, and here again I think a lot depends upon how strong analog indoctrination is on a university campus. If researchers, both students and faculty, are familiar with instrumentation techniques in general, then they're pretty capable at building bridges.*

SMITH: *In relationship to batch-processing vs. time-sharing, about how much use should be being made of a computer before it becomes economical to use the time-sharing technique?*

PAYNTER: *Your option may be partly removed by the manufacturers themselves, because within two or three years the equipment at your campus is bound to permit some modicum of time-sharing anyway and the interface here is getting less and less distinct. Most time-sharing systems now have something equivalent to so-called foreground-background where you either have it available or have it planned so that, in effect, you can run a batch from the console and this allows you to use the time-sharing console as a monitor in control for a low priority batch running. So, it seems to me that almost the converse is true. The ideal way to learn programming, if you can possibly afford it, is in real-time time-shared mode.*

MILLER: *The trend is very much in the direction of answering that question. It's not a question of how large you have to be, or how much use you have to have, before you start time-sharing in the sense of multiple access. The newer systems coming out now for computer-aided instruction right down to the high school level consist of a very small computer with a number of typewriters hung on it. One doesn't have to think in terms of a giant, or even a medium-sized computer, to have multiple access.*

SHANNON: *One comment along this line. At Dartmouth, I'm a big user and I've gotten shoved off because these thousand little users have come in. One of my students said, "I'll fix you up," and I think by next week we will be able to come in the background mode and run automatically chained programs in background. I'll be able to put in a program which occupies ten core loads of 6K each, and swap back and forth, and just come on at night and run. But I do have that functional capability within the system and this is what's going to happen in the next generation of time-sharing systems; you will be able to develop the programs in foreground, stick them in the background, give them a low priority and say, "Send me the answers in the morning."*

A LANGUAGE PROCESSOR FOR INTRODUCTORY ENGINEERING COMPUTATION

RICHARD W. CONWAY
Cornell University
Ithaca, N. Y.

The use of computers in engineering design is already impressive and the prospects along this line, with the advent of remote-access and graphical communication devices, are very exciting. The rapidity with which the importance of this technology has been recognized by engineering education so that it has been given a place in already hard-pressed curricula is equally impressive. However, I rather doubt that this time is being as well used as it might be. I think that some of the time and effort now being spent on programming instruction could be better used to give a broader introduction to computer science, and that this could be done without reducing the students' programming facility simply by employing a more appropriate language and processing system than is typically used.

In principle, these questions are really quite independent. If present instruction in programming can be improved, that is presumably worthwhile. It does not immediately follow that any time that is gained should be invested in a broader exposure to computer science. In practice, it is much easier to shift emphasis within a course than to change the time allocated to a general area.

The question of what material in computer science should be provided to all engineering students is, of course, the more important and more difficult problem, and I treat it briefly in the following only because I am less confident of my answers. I believe that the computer field is in real danger of becoming as much involved in the peculiarities of current technology as engineering in general was some fifteen years ago. Especially in this rapidly changing field there is real necessity to identify the basic concepts concerning information and its manipulation, as opposed to the details of a particular machine and a particular language. The obsolescence of these details is incredibly rapid and complete. For example, estimate how much of the material of a typical computer course of the mid-fifties involving the IBM 650 is still relevant today.

It is, of course, far easier to point out what is not fundamental than to state what is, but as a starting point I think that it is necessary to come to regard the computer as a more general device than a calculator. A student does not really understand the character of the computer until he appreciates that, in some applications, the machine is really a device for storing and retrieving large volumes of information, and requires sophisticated systems for classifying, linking and searching these files. In other applications, the computer is made to implement a special algebra, manipulating symbols according to operators that have nothing to do with familiar arithmetic processes. This is particularly important to the engineer in applications involving simulation of dynamic systems or the automation of the design process. However, current instruction is concentrated almost entirely on the calculator view of this many-faced machine. And even this is not being done very well. It is losing sight of the key points in a plethora of technical detail and taking too much time in the process.

Probably the fundamental concepts in this calculation are the assignment of value to variables and the conditional control of sequence. These are, of course, present in FORTRAN and can be properly underscored by a knowledgeable instructor, but there is a tendency for them to be obscured by the mechanics of the process. For example, the distinction between fixed- and floating-point numbers contributes nothing at all to the understanding of the basic process, nor does it add to the students' problem-solving ability. It is strictly a question of machine convenience and efficiency, yet time is spent defining and explaining these forms and conveying the timeless truth that integer-valued variables must have identifiers beginning with I, J, K, etc.

There is no question that students can learn and use FORTRAN, or more typically a subset of FORTRAN, in the time allocated in introductory courses. However, there is also no question that comparable power and understanding can be achieved by the same students in a fraction of the time by the use of a language specifically designed for this purpose. This has been proven in practice with a language called CORC at Cornell since 1962, and more recently with BASIC at Dartmouth. There are two entirely different computing worlds. One is the world of the experienced programmer who is concerned with compactness of expression, efficiency of execution, and to whom the many rules and conventions of a production language become second nature through continuous use. The other world is that of the neophyte and intermittent programmer who is struggling to learn or to remember the essential form of the language and who will sacrifice much convenience in favor of simplicity. With such different circumstances and objectives, it would indeed be surprising if the optimal design of a computing language turned out to be FORTRAN in both cases.

Similarly, the translators and monitors that are typically used for introductory instruction were not developed for that purpose and do not do all that they could for the student. The CORC processor can illustrate some of the services that can be provided, but it does not exhaust the possibilities by any means. CORC begins by scanning the source for syntactic errors, probably more exhaustively than any other such processor. For example, by examining context and usage, CORC will often conclude that an identifier is actually a misspelling of another identifier and does not represent a distinct variable. The important innovation is that CORC not only discovers errors and reports them to the programmer—it attempts to correct them. For example, it performs spelling correction by looking for transposition of adjacent characters, omission of terminal characters, concatenation errors, keypunch shift mistakes and confusion between zero and the letter "o." It supplies operators, parentheses, labels, reserved words, and even statements in some special cases. In fact, no matter how badly the source description may be written, CORC restores it to a form that is syntactically correct, generates the object code, and begins execution. The correction is attempted not so much in the hope of being able to reconstruct what was intended, as simply to produce an executable program. There is much diagnostic information that cannot be obtained until execution, so that refusing execution when errors are found in translation increases the number of machine approaches that are required and lengthens the testing process. To our surprise, the corrections often turn out to be valid and at least one pass at the machine is eliminated. I would estimate that at least one-third of the final programs submitted to me have at least one instance where CORC has successfully corrected some minor error of spelling or punctuation and the student did not find it necessary to resubmit the corrected program.

During execution, the monitor communicates with the programmer in source language terms, referring to identifiers and statement numbers. This is done, without recourse to interpretive operation, by retaining the identifiers in the symbol table and by inserting statement numbers in dummy instructions in the object code. This exacts a minor price in space and time that a production-processing system would be unwilling to spend, but it is entirely appropriate here. It means that the programmer never requires, or is confronted by, information in a foreign form. Finally, after execution is terminated (either naturally or by exceeding error count or time limits), a dump of variables and final values is automatically provided and a tabulation of the frequency with which each label was encountered during execution is given. This latter has turned out to be invaluable diagnostic information.

This processor was constructed with the assumption that this would take more machine time, but that this was a price worth paying for more effective assistance in instruction. It turned out that there was no price to pay. By recognizing that the programs for which this was intended would only require a small fraction of the 32,000-word storage of the CDC 1604, we made the compiler permanently reside in core, (and suitably protected it from errant student programs). Since the setup time for each program in FORTRAN under the CDC 1604 Monitor is 25 seconds, this meant that our average time of about two seconds for compilation and execution of these programs compared rather favorably with the 26 to 27 seconds under FORTRAN. (Card-to-tape and tape-to-printer operations are handled by a satellite 160A.) In addition, the number of machine approaches to achieve successful execution of a problem was just about halved in comparison to our previous experience with FORTRAN and Burroughs ALGOL. Finally, the system proved sufficiently adept at correcting keypunch and other spelling mistakes so that we could abandon key-verifying of programs. All of this is bonus and I would like to believe that we would still be using the CORC system even if it ran more slowly than FORTRAN.

We are aware that some of the necessity for this type of processor is eliminated if a student can write, test and execute his program in conversational mode at a terminal of a time-shared computing system. However, we are convinced that there is still a modest future for a well-conceived, batch-processing system for student programs. It is going to be some time before the luxury of time-shared computing is available to all of the institutions that must provide this introductory instruction, and, even for institutions that have large-scale time-shared installations, it is not at all clear that they can afford to be entirely terminal oriented for high-volume introductory work. I think that a good multi-programming system such as the one at Case, in conjunction with a CORC type of language and processor, would be a very effective and economical means of handling high-volume instruction. At least the fully conversational systems should be required to justify their existence against this alternative, rather than against the current typical batch-processed FORTRAN.

The same situation holds with greater strength for computer use other than numerical calculation. Languages for non-numerical and list-processing are even less suitable for introductory instruction, and have effectively limited this important type of computing to advanced students. In 1963 we expanded CORC in a manner patterned after SIMSCRIPT to a language called the Cornell List Processor. This provides a slightly more advanced student with access to a language that is convenient for such problems as the simulation of dynamic systems, game playing and heuristic programming. Programs which constituted a master's thesis before CLP are now given as homework problems in undergraduate courses.

We are currently developing an instructional version of PL/I. We have extracted a subset of statements consisting of assignment, READ, WRITE, GO TO, IF ALLOCATE and STOP and have adapted the PERFORM statement from COBOL to serve in place of PL/I's DO, CALL and PROCEDURE statements. This PERFORM modification makes our version not quite a proper subset but we believe that the resulting simplification is well worth the departure. The language goes beyond PL/I in the sense that is has fully dynamic storage allocation and direct matrix operations. For example, in X = (A+B) *C or IF Y*X = A . . . the operands can be scalars, vectors or matrices, limited only by the usual algebraic rules for conformability. This Cornell version of PL/I is being implemented with a translator and monitor that provides even more ambitious assistance than CORC. This will be available for "small" IBM 360s (Model 30 or higher, with disc and 64K of main storage) late this year. During 1967 an extended version including list-processing and non-numerical capability will be released. Shortly after Cornell installs a time-shared 360/67 in September, 1967, the language will be adapted for remote-terminal operation.

In summary, I am not trying to sell a particular language but rather to make the following points:

1. It is at least as important for engineering undergraduates to understand the fundamental character of the computational process and be able to view the machine as more than a calculator, as it is for them to achieve proficiency in programming in a particular language for a particular machine.
2. This broader objective can in part be covered during time that could be salvaged from existing courses by the use of a language and an operating system that has been specifically designed for the needs of undergraduate instruction.

References:

1. R. W. Conway and W. L. Maxwell, "CORC-The Cornell Computing Language," *Communications of the ACM*, Vol. 6, No. 6, June 1963.
2. R. W. Conway, J. J. Delfausse, W. L. Maxwell and W. E. Walker, "CLP-The Cornell List Processor," *Communications of the ACM*, Vol. 8, No. 4, April 1965.
3. W. C. Lynch, "Description of a High-Capacity, Fast Turnaround University Computing Center" (Case Institute), *Communications of the ACM*, Vol. 9, No. 2, February 1966.

PASKUSZ: *I think you're a little pessimistic as far as the time is concerned when we go to a remote console, time-shared operation. I can't really see any difference in time-per-student involvement on a console versus the time-per-student involvement on, say, a keypunch machine. I think they would be comparable. There may be a factor of two, or something like that. We have about 2,000 engineering students now and a couple of required courses which they all take; and we've also got a couple of elective courses which many students take. With the half a dozen keypunch machines we have there isn't any great amount of queueing, so I don't think that there will be a great problem of how many of these remote consoles you're going to need if you use them for basic instruction.*

CONWAY: *I agree that this will take the same order of magnitude of time. We find that it's not much more expensive to add a girl to the keypunch, and that we actually can cut down on the number of punches we would otherwise require at very little extra cost, and I think this is a reasonable thing to do for a student. After all we're not trying to teach him to punch cards; the way our operating system works, it's being done for the student on the first pass and he makes corrections. So if there is something about getting one's hands on a keypunch, the students do experience it for small bits of time. I wish I could believe that we could provide enough terminals at essentially $1,000 rental apiece to do this for the students.*

ROSENTHAL: *I'd like to make a very quick defense of FORTRAN as the introductory language. Namely, you don't have any textbook available for the other language. What I mean is an available, undergraduate, introductory textbook to introduce the system and the language. That I think is the primary prerequisite for wide use of the language. I agree that you need something other than FORTRAN, but not until you produce a textbook.*

CONWAY: *I'm not really trying to sell anyone CORC for the simple reason that most of you couldn't use it even if I did convince you. CORC was done on a shoestring by some people in the Industrial Engineering Department, and it was done over the objections of the Computing Center* *at Cornell and the Mathematics Department. It took a little less than a year before it reached the point where all introductory instruction on the campus was done on CORC. Students graduate into FORTRAN and ALGOL as soon as they have some significant computing to do, something that will run into a dozen minutes or so. I would like to make you unhappy with certain of the things that FORTRAN is and does, but I don't really have anything to offer you in the meantime, because you couldn't use CORC even if we did have all those textbooks.*

FENVES: *To answer Mr. Rosenthal's comment, with due apologies to Dr. Organick who is sitting in the audience, I maintain that there is no acceptable FORTRAN textbook available.*

MOTARD: *Listening to these various comments, I think what we're trying to do here collectively is like designing a system, and we on the team have different attitudes because we haven't really come to grips with the question of what is the unit module that we want to use as the unit of communication between the man and machine, and we have also found out that this is going to vary from one user to the next. For instance, the most sophisticated users want unlined, time-shared consoles because they've mastered the artificial languages and all the sophisticated techniques and they think now that the limitation is accessibility and feedback; the neophyte hasn't even gotten to that stage, so we have to take him by the hand and thus need a different kind of system. It's a real question of whether we are going to be able to design the ideal system to satisfy all needs, and I think we can, and we will need to have a range of responses according to the units of communication input, the time-scale response that we want, and the degree of analysis that is to be done.*

CONWAY: *I am in complete agreement. My point mainly is that the design problem for this large and important segment has been neglected in the past; that we have been given a second-hand product, we have had an adaptation of something that was never intended for this purpose. Certainly the complete system that we're designing now includes a central processor, various remote devices and a lot of language processors. I would be willing to conclude simply that it's going to take more than one language processor to serve a highly disparate group of users adequately and fairly.*

ORGANICK: *I was very excited about the very beginning of your paper in which you spoke about the three points of view of the computer and I realized you simply didn't have the time to develop the explanation of how you were going to provide, in an introductory course, these other two views. I would like to call the audience's attention to a very interesting report, Monograph 7, of the Discrete Systems Project. In this report they attempt to provide their view of how and what they would do to introduce a student to computer science. They make a rather strong point, as I'm sure you would, if time permitted, that there are at least two equally important intellectual processes in the programming of the solution of a problem on a computer. The one we have been guilty of overemphasizing to the exclusion of the other is, "How do you organize the steps of the problem that you want solved." The second very important one is, "How do you conceive of the data and its organization, first at the abstract level and then at the level of machine representation?" And I would agree with Fenves that all of the FORTRAN books written for beginners so far neglect the second aspect. I think this is where we have been weakest in our engineering introductory courses so far.*

PRECEDING PAGE BLANK- NOT FILMED

MANUFACTURERS' FORUM

SECTION E

Round-table presentation by computer manufacturers on products and concepts pertinent to the use of computers in engineering design.

CHAIRMAN: S. G. CAMPBELL, *Assistant Vice President,*
Xerox Corporation, Rochester, New York

AUTOMATED DESIGN WITH HONEYWELL SYSTEMS

R. Roderick Brown
Honeywell, Incorporated
Waltham, Mass.

The use of computers in engineering design has proliferated rapidly in the last decade. The dominant characteristic of this growth has been the increasing number of different applications to which computers have been put. Electrical apparatus, communications and power systems, structures, control systems, and other topics discussed at this conference, have all evolved in practical, computerized form only recently.

With but few exceptions, each of these applications shares a common attribute: "single-mindedness." That is, each program has been designed to solve but one aspect of the overall engineering problem. For example, the electrical-apparatus design program might calculate the performance of the device, but would provide no directly usable information on its fabrication. In each case, the primary objective has been to solve the outstanding problem: to reduce the burdensome calculation or improve the accuracy (or speed, or cost) in accomplishing a large number of repetitious operations.

As each engineering discipline fortifies itself with this essential tool, a trend is emerging towards a broader view of the application of the computer. This view, which may be termed the "systems" approach, recognizes the role which the computer can play in the conduct of an industry's business. The concepts of the "total systems approach" and "integrated management systems" are now being vigorously pursued in the areas of financial and production control, where solutions to the outstanding problems have been achieved, and the integration of single-minded programs into a cohesive structure is being implemented. These same concepts are now entering the engineering domain; the future will see their extension across the organizational barriers between marketing, engineering, and production.

One of the more prominent examples of the reality and practicality of this approach exists in the area of the engineering design of digital computer systems. An example taken from Honeywell's experience as a manufacturer will serve to illustrate this trend in action. Prior to 1958, the application of computers to aid in their design was sporadic and rudimentary. At that time, two applications were developed, one for logic design analysis, and one for the design of wiring suitable for fabrication by a numerically controlled machine tool. In 1960 these two applications were linked together and a materials-explosion application was initiated. In 1961 the statistical analysis of component failures was placed on the computer, as was the design for fabrication of all wiring. In 1962 the utilization of computers for production applications, such as inventory control and the maintenance of service records, reached significant proportions, and in 1963 and 1964 these areas were linked to the engineering applications, which had also been interconnected by then. During this time, the use of the computer by individual engineers solving local problems amenable to formulation in FORTRAN also increased, primarily in areas which had not been previously exposed to computer aids. In parallel with these activities, other areas of the company were continuing the development of employee, financial, and marketing-program systems. As would be expected, the latter did not influence the engineer directly except through the competition for a share of the limited computer time available.

From this experience, and from that of other users of computers, we can draw several significant conclusions. Most apparent is the evolutionary nature of the application of the computer to design. Within an area of an engineering discipline, the first contact with the computer is a very personal, individual affair, concerned only with a very limited but highly irksome problem. The initial discourse is very awkward, partly because the problem must be translated into a language which already exists but is not truly suited to the situation, and partly because the engineer must express his problem to another person (the programmer). In either case, the engineer will suffer some initial frustration because he will not have expressed his problem with sufficient preciseness to satisfy the requirements of a computer. With time and patience, however, he will be rewarded with an answer. In this way, the programs which I have called single-minded are created. If they are successful, they will be used over and over again, and will become a working tool in the design organization's repertoire.

When there are several such programs in existence in an area, the thought comes to mind that they should be integrated into a "total system." This is economically feasible only when approached in the proper manner. The term "linked" has been used above to describe the evolution of some phases of design at Honeywell. By this is meant a joining together, as two islands might be joined by a bridge. The analogy is apt, for the traffic (data flow) across these links is usually aperiodic, independent of the existence of the link, and usually restricted to a small portion of the population. This is particularly true when different engineering areas (such as electrical design and production engineering) are linked, for each needs only a distillation of the data required by the other. To provide more would be uneconomical (higher transmission or storage costs) as well as chaotic (note that the link provides information, rather than data, which is quite different). These links are economically justifiable because the information captured in one area is made available to the other without requiring a man-machine interface with all of its attendant problems.

Within a given area of engineering design, the links should be much stronger, and may even approach full integration, wherein data flows are frequent, automatic, and prolific. Such integration is necessary when large and very complex designs are produced by big groups of people working together. Here data once captured is of utility to all concerned and, indeed, the computer performs a true service by providing a uniform representation of it to all who must work with it. When these single-minded programs are integrated, we also see the major emphasis on the development of a problem-oriented language for communicating with them. The objective of such a development is clearly to remove the frustrations which plagued the early development of the programs, and which linger on; and to improve the efficiency of describing problem statements by increasing preciseness and conciseness and eliminating the unessentials.

Another major conclusion which can be drawn from our experience is the direction in which the engineer's working environment is changing. Initially, the designer using computers does so on an occasional basis. Only a part of his time is spent getting a solution to that special problem, while others less difficult are done by hand; and, of course, he must communicate his results to others and in turn receive their critique of his design. At this stage, the computer is a tool, off in a corner out of sight, out of mind. With the linking and integration of the programs used by the company, the computer now becomes an integral part of his working environment. It is used daily, aiding in the accomplishment of design and automatically providing the communication links to other areas. If he was a good designer before, he now has less such work to do; if he was mediocre before, he must do more, but the quality of his results is increased. This latter results not because the computer can necessarily produce good designs, but because it can evaluate their quality and report to the chief designer more than was heretofore possible. Thus the engineering designer finds himself relying on the computer in his work as much as he does on his car to get him to work. He could get along without it, but would rather not. The nature and quality of his work is known better by others, and recognition and reward based on more knowledgeable evaluation is possible.

One further point must be made. It is the contention of this writer that the evolutionary nature of the design process using computers and the changing environment of the engineer represent a pattern which we shall see repeated many times in the future. Each major new idea in a discipline, each technological breakthrough, will start the entire process anew. The transitional phases we have described will be more easily and quickly accomplished, but they will be there.

What does this all mean to the colleges and universities which teach our engineers and to Honeywell as a manufacturer of computing systems? Each in our own way must be prepared to cope with a wide diversity of problem situations. Versatility, adaptability, and high capabilities must be the prime attributes of our products, both men and machines.

AUTOMATED TECHNICAL DOCUMENTATION SYSTEMS AND ENGINEERING DESIGN

PAUL H. ROSENTHAL
Univac Division
Sperry Rand Corporation
El Segundo, Calif.

The current pace of technological development is causing an information explosion that is severely taxing current documentation methodologies. The automated man-machine, engineering-design systems being discussed at this conference will not reach the performance levels hoped for unless the documentation bottleneck is broken. This paper will therefore outline some of the automated-documentation system work being performed at the Univac Division of Sperry Rand Corporation as a portion of the total effort in the engineering design area.

The hardware and software systems currently being marketed and installed by the Univac Division will allow the full facilities of a computing center to be utilized from the desk of the engineer or executive. A limited automated-documentation system is currently in pilot test for those hardware systems which will process the mass of documentation produced by our development groups. Later on, extensions of this system will give it the capabilities needed by complete man-machine design systems. Figure 1 outlines the capabilities of this initial documentation system. Current information-systems methodologies are used by the system to automate software and hardware equipment manuals. These manuals are subject to constant revision and are distributed widely both inside and outside of the company.

UNIVAC AUTOMATED DOCUMENTATION SYSTEM

UNADS

***Maintains Documents on Magnetic Tape Files**

***Inputs Text and Corrections**

***Outputs Proof Listings and Final Documents**

***Performs Line and Page Composition**

Figure 1.

Figure 2 is a schematic of the flow of information in the computer system. Current input is via paper tape or cards, usually from a remote site. The primary output is ä proof listing containing the text of the document and an index. It is now produced on a high-speed printer at the remote side. Extensions to the current system will, of course, incorporate remote typewriters for input and output. The documentation systems runs on the UNIVAC 1108 Computer,[1] which is a large-scale multi-processing system designed for computer utility operation.

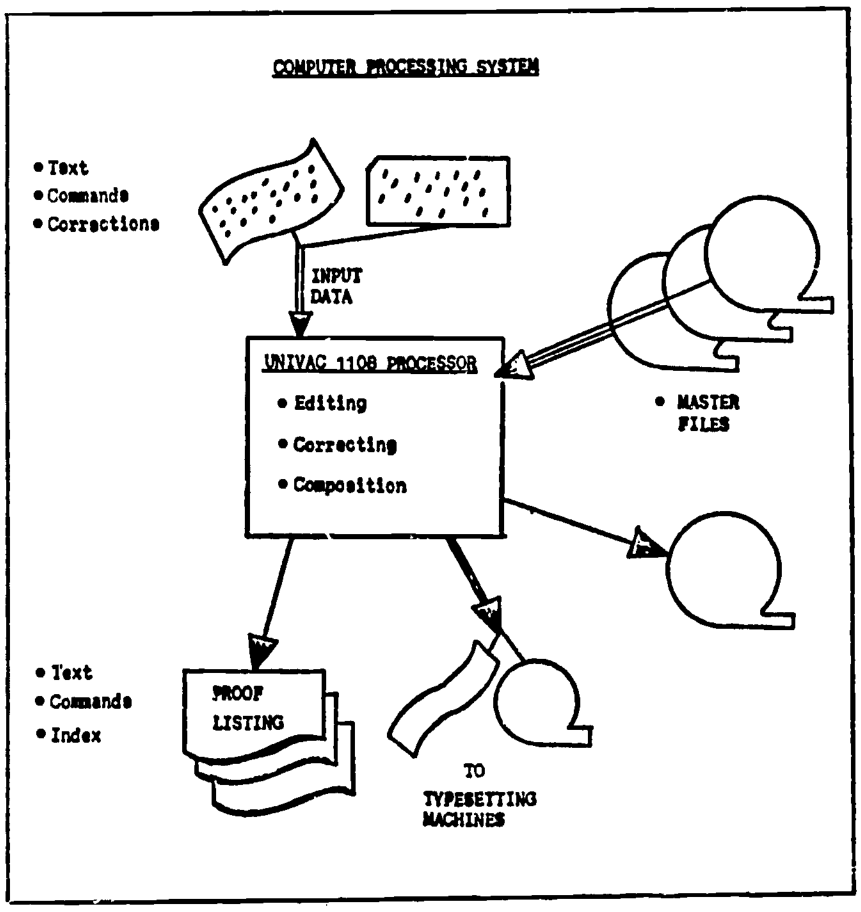


Figure 2.

Figure 3 defines the two types of input data to the documentation system. Commands are separated from text by a starting and ending command-delimiter code that is specially assigned to each type of input unit. In the examples that follow, left and right brackets will be used to represent these delimiters.

INPUT DATA TYPES

TEXT
***Information to be printed on Final Document**

COMMANDS
***Control Composition**
***Control Correcting**
***Information needed for UNADS Program**

Figure 3.

[1] UNIVAC is a registered trademark of the Sperry Rand Corporation.

Proof Listing

Text	Index	Commands
THE TEXT MATERIAL	24 1	0 PAR
IS LISTED, COMPOSED	24 2	
AND HYPHENATED AS	24 3	
IT WILL APPEAR ON THE	24 4	
FINAL DOCUMENT.	24 5	
COMMANDS ARE EX-	24 6	0 PAR
TRACTED AND SHOWN	24 7	
IN RIGHT COLUMN.	24 8	

Figure 4.

Figure 4 illustrates a proof listing. Since most remote- terminals do not have upper/lower-case printers on-line, we slash the characters to represent an upper-case character. Except for justification, the proof listing simulates the final document as closely as possible. The index shown is used for correcting and editing the text and commands shown. Correcting can be done at the section level (illustration at 24), at the line level (illustrated at 1 through 8), or at the word level. The commands shown are preceded by the word number which they follow. They are removed from the text and listed to the right to improve the proof-listings simulation of a final document.

The data stream through the UNADS systems consists of data and commands. Figure 5 lists the four types of commands and an example of each.

COMMAND STRUCTURE

BASIC COMMANDS

*FORMAT [INDENT, LEFT 3]

*ACTION [LINFED 3]

*EDITING [ADD, AFTER 27-3-4, "WITH"]

*INFORMATION [INLIB, 2457, "PERT MANUAL REV. 1C"]

Figure 5.

Format commands modify the composition of text from the point of entry until they are modified. The 'INDENT' command shown will move the left margin 3 spaces (in printing nomenclature 3 em spaces) to the right. Typical format commands include one each for page size, a margin set, line spacing, a column set, and a tab set.

Action commands modify composition at their point of entry but have no lasting effect. The 'LINFED' command illustrated will terminate the current line being composed and move the paper or film 3 spaces (in printing nomenclature lead 3 ems) vertically. Typical action commands include those for pagination, graphics, tabs, quadding, and flushing.

Editing commands control the correction and merging of master files with data from the input stream. The 'ADD' command illustrated will insert after word 4, line 3, section 27 of the current master file the word "with." Typical editing commands include the 'COPY' command, and delete and revise. A special command, 'REPLACE,' will search the master file for a phrase of text and replace it with another phrase. This allows selected text to be tailored for a product subsystem by changing each occurrence of specialized specifications or names.

Information commands supply parameters to the UNADS program and do not have any visual effect on the composed output document. The illustrated 'In Library Tape' (INLIB) command contains the information needed by the program for checking the label of the magnetic tape being used as input. Information commands specify parameters such as change letter, end of data, and input/output machines being used.

COMMAND STRUCTURE

USER DEFINED COMMAND

*DEFINING

[PAR=LINFED; TAB] [CHAP=SECTN; PAGE; RESTOR; LINFED, 6; HA; QUAD, CENTER]

*USAGE

When Starting Paragraph	[PAR] The . . .
When Starting Chapter	[CHAP] Introduction . . .

Figure 6.

The entry of the format and action commands which control composition can be very redundant. Long streams of commands are required to define the start of chapters, sections, and paragraphs. Each type of document will require extensive set-up for page formats. The redundancy problem is solved by allowing the user to define his own commands. Figure 6 illustrates the defining and usage of user-defined commands. At the start of a document all of the standard definitions to be used in the document are entered, thereby substantially reducing the typing requirements.

Figures 7 and 8 illustrate the contents of some of the composition and editing commands. The complete set of commands include the capability for directing most of the machines which are used for the production of printed matter.

SAMPLE COMPOSITION COMMANDS

```
PAGSIZ, WIDE 6, HIGH 11
IDFORM, TOP .5, FACING
GRAFIC, HIGH 3.5, WIDE 3.5, LEFT, TOP
QUAD, CENTER
FLUSH, ".".
```

Figure 7.

SAMPLE EDITING COMMANDS

```
COPY, 2457, FROM START THRU 121-48
REVISE, 21-7-5 WITH "REGISTER"
REPLACE, "DOW JONES" WITH "A.M.C."
ADD, AFTER 120-38-4, "RESULTING,"
DELETE, 120-42-7
```

Figure 8.

A diagram of the hardware configuration used by the UNADS program is shown in Figure 9. The UNIVAC 1108 processor will run the program in time-sharing and multi-processing modes. Input data normally is received from a remote paper-tape reader. The input and output master files are handled by tape units at the computer site.

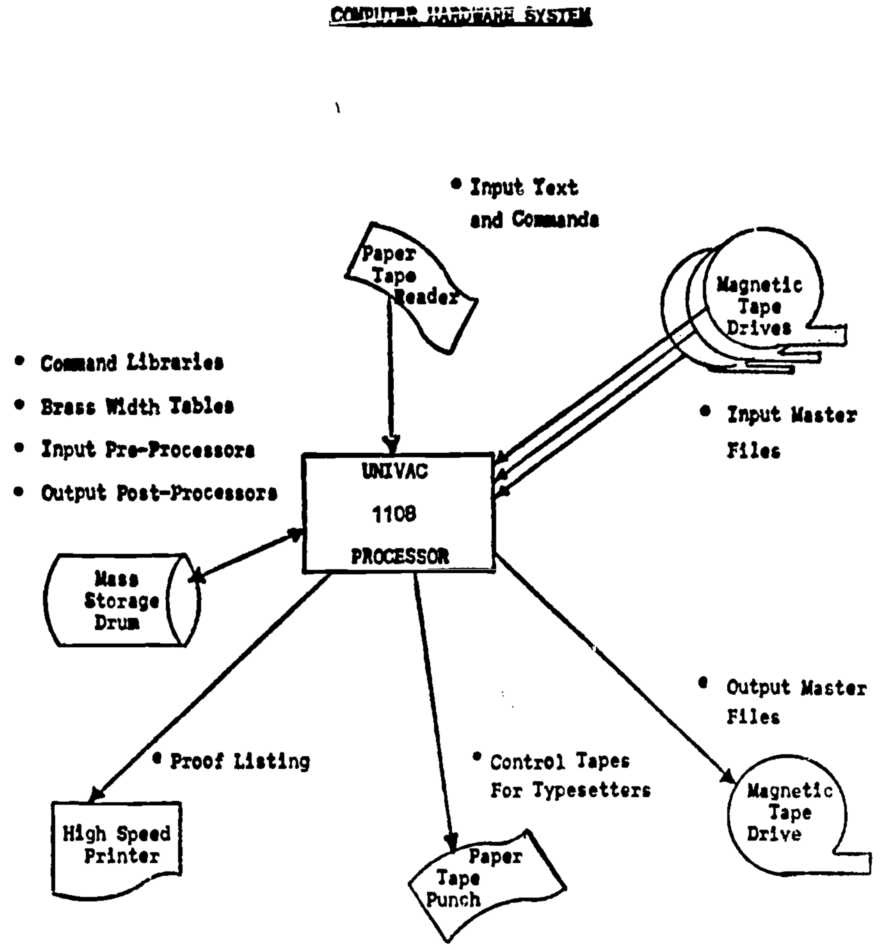


Figure 9.

The mass storage drums on site contain pieces of the UNADS program called by individual jobs. Command libraries are groups of user-defined commands defining a specific document type and typing format.

A command library entry can be entered into the input stream simply by giving its name within delimiters. Brass width tables give the size of the characters in each of the various type fonts available on the typesetting machines. An input and output processor exists for each kind of standard input-output equipment available to individual users. They translate specialized external formats to the standard format used by the computer internally and stored on master files.

The proof listings and control tapes for printing final documents are produced on high-speed printers and paper-tape punch located at the remote site. Figure 10 is a photograph of the UNIVAC 1108 computer used by the UNADS system. It is a very high-speed, large-scale system currently in use with remote terminals. The UNADS system is being coded and tested in Los Angeles through a remote link with our department's UNIVAC 1108 computer in Minnesota.

Figure 10.

The UNADS system is being developed in three phases. Phase I is designed for the needs of our own documentation problem. It will be used to produce the documentation for our hardware and software systems. Phase II incorporates a wide range of pre- and post-processors for usage by a wide variety of computer users. Phase III incorporates remote correcting and editing by the console of master files contained in data banks. This phase meets all the requirements for documentation support of automated engineering-design systems. Figures 11, 12 and 13 list the features of these phases and demonstrate the eventual scope of the project.

FEATURES UNADS I

* SINGLE COLUMN PAGE COMPOSITION
* TABULAR MATERIAL COMPOSITION
* AUTOMATIC TABLE OF CONTENTS
* MAGNETIC TAPE MASTER FILES
* LINE JUSTIFICATION WITHOUT HYPHENATION
* INPUTS — PAPER TAPE, CARDS
* OUTPUTS — TYPEWRITERS

Figure 11.

FEATURES UNADS II

* GRAMMATICAL HYPHENATION ALGORITHM
* LINECASTER POST-PROCESSOR
* PHOTO-TYPESETTING POST-PROCESSORS
* AUTOMATIC FOOTNOTING
* AUTOMATIC PARAGRAPH NUMBERING

Figure 12.

FEATURES UNADS III

* REMOTE EDITING
* MULTI-COLUMN PAGE COMPOSITION
* AUTOMATIC INDEXING THROUGH THESAURUS
* MASTER FILES ON DATA BANK or MAGNETIC TAPE

Figure 13.

The Univac Division of Sperry Rand Corporation will probably use UNADS Phase I for approximately one year, gaining experience and modifying the system for optimum productive performance, before UNADS Phase II is released to the field. We are just now starting our training of personnel for usage of Phase I and hope to be in production in the near future. We hope that the UNADS project and others of its kind will appreciably reduce the cost- and time-scales of documentation currently caused by the information explosion.

COMPUTER CONTROL COMPANY PRODUCTS AND THE ENGINEERING DESIGN ENVIRONMENT

R. A. COWAN
Computer Control Company, Inc.
Framingham, Mass.

The use of computers as an aid in engineering design is a step toward the development of an integrated complex of men and machines. The purpose of this complex is to use the creative and imaginative powers of the man and the analytical and computational powers of the computer to efficiently carry out the iterative process of creative design. The key to the use of computers in the design process is communication between the designer and the computer.

To fulfill the needs of the engineering design application, the computer should have a comprehensive instruction repertoire and flexible high-speed I/O (Input/Output). It must be easily expanded to provide for increases in the requirements of the engineering designer. It is especially important that it have graphical I/O capabilities such as CRT displays or plotters. It should also possess hardware interrupts for use by the designer in communicating with the computer and for providing efficient I/O communication with the peripheral devices. The software associated with the engineering-design computer system should be designed to simplify the process of stating the problem and should minimize the tasks of set-up and modification of design parameters. It is desirable that the computer should be valuable to the neophyte programmer as well as the expert. For this purpose, higher level languages such as FORTRAN are necessary to simplify the programming process. A supervisory program should be provided, structured in such a way that either modification, or addition, to the design programs is possible in a simple fashion. In this way, the designer can build upon previous programming and designing experience and expand his capabilities.

Computer Control Company (3C) presently produces three digital computers which are applicable to the engineering-design field. They are the DDP-124 and 224, both 24-bit computers, and the DDP-116, a 16-bit computer. These computers are all modularly designed, high-speed computers with parallel organization. They have a variety of I/O capability and are easily expandable to increase the capacity of the computer. A range of peripheral equipment is offered with them, plus many internal options. As standard equipment, they include indirect addressing and indexing capabilities and priority interrupt schemes. The I/O bus system is used, which is extremely flexible and allows ready expansion of the I/O capabilities. The basic software package includes the DAP assembly language program, FORTRAN IV (ASA FORTRAN Standard), and executive routines. Also included are a selection of utility programs for aiding the programmer in debugging, a complete I/O library and a mathematical library.

DDP-124

The DDP-124 is a fast, general-purpose digital computer featuring μ-PAC monolithic, integrated-circuit construction. The standard mainframe includes 8K of memory with a 1.75 microsecond memory cycle, hardware index register, fast multiply/divide hardware, electric input/output typewriter, 300 cps paper-tape reader and 110 cps paper-tape punch. It is fully program-compatible with the DDP-224. Modular design and broad I/O options enable the DDP-124 to be tailored to a variety of applications on or off line. The standard mainframe is competitively priced at $65,000.

The flexible 124 command repertoire works with sign-magnitude arithmetic and contains 48 instructions. It includes, for example, logical commands, an executive command and a step-multiple precision command for very simple and fast multiple-precision routines. The jump-storage and jump-return commands provide easy subroutine linkage and nesting of subroutines. Also provided are a variety of conditional program-transfer commands.

The combinations of input/output capabilities that are provided are specifically adapted to those applications which require fast input/output data transfer and easy adaptation of the computer to allow efficient operation in many different system configurations. For this, the DDP-124 provides a variety of input/output channels, four different input/output modes of operation, easy modular expansion of its I/O capabilities and high data-transfer rates. Data may be transferred directly either into or out of the main arithmetic register or any memory location with a single command. Transfer of character inputs or outputs into or out of the arithmetic register can be controlled by a mask included with the instructions. Any combination of up to 6 bits of the character data can be read or generated to allow flexible formatting. Input/output rates of up to 285,000 24-bit words per second are possible using the standard input/output commands.

Any input/output channel can be operated in either Ready Mode or Interrupt Mode. In the Ready Mode, the computer program tests periodically for channel-ready signals. If the test is successful, the program will transfer to the proper subroutine; if not, the program carries on with the current routine. Simultaneous testing of up to 12 I/O channel-ready signals is possible. In the Interrupt Mode, the computer program is automatically interrupted and a program jump takes place to the memory location directly related to the interrupt concerned. On completion of the interrupt subroutine, the main program resumes at the interrupted point. Single bits of input or output information may be transmitted via the sense lines or output-control signals respectively.

The Direct Memory Access is an optional input/output mode for direct access into or out of successive memory locations, independent of the computer program after being initially set up. The DMA channels have priority over the control unit of the computer for memory access, and will steal cycles from the processing of the computer at rates determined by the devices being serviced. Data rates of up to 570,000 words per second are possible.

The standard 8,192-word core memory is readily expandable to 32,768 direct-addressable words, 64 sense lines and 64 output-control pulse lines. Also available are character input/output channels.

DDP-224

The DDP-224 is a high-speed, 24-bit, general-purpose computer constructed with S-PAC digital-logic modules. The DDP-224 includes all those features which are available on the DDP-124, plus those features described below. A standard mainframe includes 4K of memory with a 1.9 microsecond cycle time, three hardware index registers, floating-point arithmetics and input/output typewriter, and is priced at $96,000.

The DDP-224 instruction repertoire includes more than 70 commands. In addition to those commands included in the 124 repertoire, it includes absolute-value add and subtract commands, normalize and scale instructions and the repeat command which allows table-look-up, block load and many other powerful operations. For data transfer from memory at very high rates, the dump-memory block and the fill-memory block commands are available. They allow asynchronous input or output of data from a block of successive memory locations controlled by ready signals from the external system. Also available is a comprehensive set of floating-point instructions as standard equipment on the DDP-224. Some typical execution times are ADD 3.8 μsec., MULTIPLY 6.5 μsec., and FLOATING-POINT MULTIPY 10.3 μsec.

Input/output rates of up to 325,000 words per second can be handled in the DDP-224 by using the fill-memory block instruction. By using the Direct Memory Access option, transfer rates of 285,000 words per second are possible in the normal mode of operation, or 525,000 words per second in the hog mode of transfer. Data transfer rates of up to 525,000 words per second are possible using the fully buffered channel option. The FBC permits fully buffered data transfers to occur simultaneously with program execution whenever a fully buffered channel addresses one memory module while the control unit of the computer addresses another module.

Special options make it possible to combine several DDP-224's into integrated, large-scale computer systems with functionally common and/or private memory, control, arithmetic, system input/output facilities and peripheral equipment. These options are the fully buffered channel, the access-distribution unit and the time-multiplex unit. The access-distribution unit permits processors and/or fully buffered channels to share a memory complex. It also contains the logic and priority system to the common memory and the distributing system for data flow to and from the processors and common memory. The time-multiplex unit permits multiple processors to communicate with a common set of peripheral devices.

DDP-116

The DDP-116 is a 16-bit general-purpose computer with a 1.7 microsecond memory cycle time and a standard 4,096-word core memory. It has a fully parallel machine organization, uses two's complement arithmetic and includes both indexing and multilevel indirect addressing. Standard features include a flexible instruction repertoire of 61 commands, a powerful I/O bus structure, a standard interrupt line and a keyboard and paper-tape I/O unit. The DDP-116 is priced at $28,500.

The command structure of the DDP-116 is unusually powerful for a machine of its size. Included are a three-way branch instruction, an increment memory and skip on zero, plus a full set of arithmetic logical and rotational shifts. The basic input/output commands test the status of I/O devices before transfer. If the test is not successful, the next instruction is executed. If the test is successful, the data is transferred and the next instruction is skipped. Also available are optional hardware multiply/divide commands. Some typical execution times are: ADD 3.4 microseconds, MULTIPLY 9.5 microseconds, DIVIDE 17.9 microseconds, interchange memory and ADD 5.1 microseconds and increment replace memory and skip 5.1 microseconds. Most non-memory reference instructions are carried out in one memory cycle.

The standard DDP-116 core memory is optionally expandable to 32,768 words. The memory is divided into sectors of 512 words each and a sector flag is used to control which sector is addressed. When the sector flag is a one, the address portion of the instruction refers to the same sector as that addressed by the program counter. When a sector flag is a zero, the address portion of the instruction refers to sector zero. Thus, 1,024 words can be addressed by a single-word instruction. Access to additional sections is obtained by either indirect addressing or indexing.

The basic I/O system of the 116 consists of a general input/output bus. This bus is used to transfer full words in and out of the computer. In addition, it contains lines which provide timing and command signals to peripheral devices. Each peripheral device tied to the I/O bus has its own buffer and control logic. This feature permits a high degree of flexibility in using multiple devices concurrently and in handling multiple devices through priority interrupt. The four basic modes in which data can be transferred back and forth between peripheral devices and the 116 are single-word transfer, single-word transfer with priority interrupt, direct-multiplex channel and direct-data channel.

The direct-multiplex channel permits data transfer between peripheral devices and the memory concurrent with computation. In this mode, the starting location to which the block of information is to be transferred and the final location at which the block transfer is to be terminated are set up under program control. The data transfer is then initiated by the program. Once this has been done, transfers occur independently of the program control until the block has been filled. Data transfers can occur at a maximum word rate of over 145 kilocycles with this mode.

The direct-data channel (DDC) is a high-speed channel capable of achieving input/output transfer rates in excess of 500,000 words per second. It can operate in a time-sharing mode with the basic computer. Program control of the DDC is accomplished entirely through the device with which it is interfaced.

SOFTWARE

A comprehensive software package which will supply the needs of either the experienced programmer or the occasional user is supplied with the 3C computers at no extra cost. The programming languages and utility routines have been designed to conform with the most widely used programmer standards and conventions. The same high degree of quality that is in the Computer Control hardware is provided to the user in the software. All software programs for the 124 and 224 are fully compatible. The DDP-116 assembler and compiler have been specially designed to eliminate the concern of the programmer for memory sector boundaries.[1] The sector boundaries are accounted for by the compiler or the assembler, and the proper inter-sector linkages established by the desectorizing loader.

A FORTRAN IV compiler is provided which complies with ASA standards, including Boolean capabilities and complex and double-precision arithmetic. A real-time FORTRAN IV compiler with recursive call subroutines is provided for the DDP-124 and DDP-224 computers with three index registers. FORTRAN IV object programs can also be run on the DDP-116 in real-time applications, provided that recursive calls are not made to standard subroutines. The DAP symbolic-assembly program which is provided conforms with SHARE standards and accepts all basic machine instructions, plus extended machine and pseudo operations. Object output may be either relocatable or absolute. Subroutine linkage is provided which will allow FORTRAN IV compatibility. On the 24-bit computers, the DAP program includes MACRO capabilities.

The utility programs supplied include a debug program which works through the typewriter, dumps, loaders, updating programs, a comprehenhive I/O library and a mathematics library. An executive-control program is available for the 24-bit computers which interprets commands through the typewriter and can load selected programs from magnetic tape. In those configurations with three or more magnetic tapes, load-and-go capability is available. A Real-Time Monitor will soon be available which will allow the user to share the computational capabilities of the computers between several programs in memory. The Real-Time Monitor will schedule and control the execution of all programs and also control all I/O operations and interpret interrupts.

PERIPHERAL EQUIPMENT

Computer Control Company offers a selection of peripheral equipment which will fulfill the needs of most engineering design situations. The standard peripheral devices offered are:

300 cps paper-tape reader
110 cps paper-tape punch
200 card/min card reader

[1] C. W. Walker, "Programming The Compacts," *Datamation*, April, 1966, pp. 31-34.

100 card/min card punch
300 line/min printer (120 column)
300 increments/sec digital plotter
Dual and triple density magnetic-tape units

In addition to the standard equipment just described, 3C is able to offer a wide range of special equipment through the facilities of the Systems Engineering Department. In this department, computer systems and related equipment are custom-designed to meet special customer requirements. These special designs have ranged from simple modifications to standard peripheral devices, and also to complex multiprocessor configurations such as the Apollo Mission Simulator. Systems Engineering provides those devices which are not yet offered as standard equipment on 3C products because of diverse customer requirements. Typical of this work has been the design of several different communication interfaces for linkage with either high- or low-speed lines in large-scale communication network and time-sharing computer centers.

Computer Control Company has also designed several different cathode-ray tube displays with speed ranges from 10 to 100 microseconds per point. These displays have included such features as character generation, line and circle generation, size control, light pens, buffered memory for refreshing of the display and keyboards. Each display is designed specifically to meet the individual customer's requirement.

The previous description provides a brief glimpse of the present 3C product line. These products, combined with the special Systems Engineering capabilities, allow 3C to meet the requirements of the engineering design field.

FUTURE PRODUCTS AND TRENDS

Computer Control Company entered the digital product field primarily as digital module suppliers. Through continuing research and development have come many improvements in 3C products such as the highly reliable and versatile S-PACs. The new monolithic, integrated-circuit μ-PACs are a direct result of a continuing research program being carried out in our Micro-Techniques Laboratory. With that line of versatile logic modules, 3C has developed the memory product lines and many special hardwired systems. From this base, we have developed our digital-computer product line and the capability to design and build large-scale complex computer-based systems. This complete digital capability provides 3C with an effective means of developing new products. In addition, this capability facilitates the expansion or modification of 3C equipment with compatible logic and memory modules directly by the customer.

Within the near future, 3C will expand its computer product line with more integrated-circuit computers which will offer significantly more computations per dollar than the present day discrete-component computers. In addition, more and lower-cost peripheral devices will become available as standard equipment, including bulk storage devices, such as discs or drums, and display units. The memory speed of these new computers will be in the 1-microsecond range. In the software area, more user-oriented application packages will be provided, in addition to the continuing 3C effort to improve the quality and capabilities of the general-purpose software programs.

The industry trends are definitely pointing toward extremely high-speed central processors. Memory speeds within the range of 100 to 200 nanoseconds should be obtainable within a few years. The size of the central processor will be insignificant compared to the equipment around it. Minus the power supply, memory, and I/O equipment, the central processor could be as small as a cigarette package.

Communication between man and the computer through the peripheral devices has not seen comparable advances in the state-of-the-art compared to the advances in central processor designs; however, we at 3C feel that this situation will change. More emphasis will be placed upon increasing the efficiency and lowering the cost of the peripheral devices when the cost of the central processing unit approaches a small percentage of the total system cost. Two areas which show high promise for tomorrow's communication with the computer are oral communication and pattern recognition. Another area that will show marked advances is the area of higher-level languages which work directly with the English language and have the feature of self-modification to improve their capabilities.

THE COMPUTER AS A DESIGN TOOL

J. Worthington
Data Processing Division
IBM
White Plains, N. Y.

The 1960s will probably be remembered as the decade in which the computer came of age as an engineering-design tool for the broad spectrum of problems. The papers presented at this conference are proof that this is true. In projecting trends, it is desirable to consider those that have brought the computer to the forefront as a design tool.

One essential is the availability of computers at a price that the buyer considers reasonable. One means of calibrating the worth of a computer offered today is to make a comparison with what was available ten years ago. Today's machines can execute at least ten times as many instructions per unit of time as did the 1956 machines, and they sell for a comparable price. Machines with arithmetic capability equal to the 1956 systems are available for about a tenth of the price of the earlier systems.

The buyer not only can get more capability per dollar, but also has many more options in size, speed, and accessories today. In 1956, the number of options offered was a little reminiscent of Mr. Henry Ford's famous comment that the buyer could obtain a Model T in any color that he wanted as long as he selected black. The Model T era of computers is past. The number of sizes of computing systems and the options available for features and accessories is somewhat overwhelming. In April, 1964, IBM[1, 2] announced its new product line which is based on solid-logic technology.

The product line at announcement consisted of six different models of processors with 15 options of memory which offered 19 different configurations. Since that time additional models and memory options have been announced, bringing the number of models to nine and the configuration total to 34.

The smallest member of the line is a low-cost, compact computing system, listed as Model 1130.[3] It is designed for general-purpose computing and is particularly oriented towards operation by the individual requiring the problem solution. The input/output equipment attachable to the system includes a typewriter, paper-tape reader/punch, magnetic disk storage, card reader/punch, line printer and a plotter. The capability offered in the system is comparable to the large machines which were available a decade ago.

The Model 1130 and its associated support combine to form an excellent tool for an engineer. The price is such that it can be justified by a small engineering organization, or by larger organizations with systems located at the points of usage. Our experience indicates that this will be a popular machine and will enjoy broad usage in engineering design.

The largest system in the product line is the System/360 Model 91,[4] which by any measure is a large machine. It is particularly well suited to solving problems that would require an excessively long time to solve on lesser machines. Because of the size and power of the system it is attractive to organizations with large problems and/or a large number of people who are solving problems.

The Model 1130 computing system and the Model 91 represent opposing strategies for satisfying computational requirements. The small machine can be effectively used by an engineer sitting at its console and interacting with it as the problem is solved. The largest machine, because of its relative speed with respect to the human reaction time, is more suited to having the engineer work separately from the system, either through batch processing of work, or through suitable multiprogramming.

Some users desire a system which offers large machine performance, and the close man-machine coupling of the small machine without undue loss of system capacity. The Model 67 Time-Sharing System[5] has been designed to service interactive, or conversational, terminals which give an engineer the full functional capability of a large machine at a response rate commensurate with his reaction time. Furthermore, it will handle batch processing as background work. The machine's attention is focused on the active terminals so that each user receives reasonable service. The time not required for servicing the active terminals is available for batch processing. Another essential to easy utilization of computers by the design engineer is the availability of terminals or transducers which facilitate the man-machine communication. The more popular interactive terminals are typewriters and cathode-ray display units. The major

[1] G. M. Amdahl, G. A. Blaauw, and F. P. Brooks, Jr., "Architecture of the IBM System/360," IBM Journal of Research and Development, April, 1964.

[2] IBM System/360 System Summary, A 22-6810-6.

[3] 1130 System Summary, A 26-5917.

[4] IBM System/360 Model 91 Functional Characteristics, A 22-6907-0.

[5] System/360, Model 67 Time-Sharing System.

trend in these devices is towards improving their function and packaging.

The utility of a terminal is principally determined on a systems level where the combination of programming support, the processor and the terminal determine the capability available to the user. As an example, a cathode-ray display device attached to a processor is quite worthless except when programmed. A FORTRAN compiler with graphic extensions can transform this combination of hardware into a rather capable tool for the design engineer.

Perhaps one of the most important ingredients in hastening the general acceptance of computers is what might be termed the macro- or application-oriented level of programming. To program a computer either at the symbolic-language or at the algorithmic-language level requires that the user must already possess the necessary knowledge about the machine and the programming language. By inserting an application language on top of the hardware and algorithmic compiler, it is possible to greatly reduce the level of knowledge requirec to utilize the computer. A good example of this is found in the coordinate geometry processor, COGO,[6] developed by Professor Miller.[7] The engineer who uses COGO needs little or no knowledge of computing and can solve problems on a computer with quite nominal instruction. The emergence of application-oriented languages represents one of the major trends occurring in the use of computers.

The multiplicity of computers and options in which they are available gives rise to a new design and selection problem. The selection of a computer involves matching user requirements against many alternate configurations. In a sense, the new freedom afforded by many options has a price, but it is offset by being able to optimize selection based on the user requirements.

From the foregoing, it is evident that computers have arrived in the engineering world. It is difficult to imagine an engineer who, in 1970, will not have a direct involvement with computers for extending his ability. The emphasis in education must be to expose engineering students to the potentials of the computer and the techniques for its usage so that they may in turn build upon the knowledge that has been set before them.

[6] COGO Application Program 1130, H20-0143.

[7] "Computing Report," The Scientist and Engineer, Vol. I, No. 3, December, 1965.

ANALOG AND HYBRID COMPUTERS IN EDUCATION IN ENGINEERING DESIGN

G. G. MOSER AND P. E. HUBER
Electronic Associates, Inc.
Princeton, N. J.

INTRODUCTION

Other papers in this session have placed emphasis on the value of digital computers in engineering education. Although Electronic Associates, Inc., does manufacture digital computers, this paper will briefly highlight the equally important role of analog and hybrid computers in education in engineering design.

A report by the committee sponsoring this conference says, "Familiarization with techniques for using both Analog and Digital computing equipment as a simulation tool can largely be integrated into the existing course structure of the curriculum." The ASEE in its report on Computers in Engineering Education makes the following recommendation: "A recommended lower-division introductory computer-use course for digital, analog, and hybrid machines is needed which should be equally satisfactory for students entering all engineering disciplines."

These direct quotes from the reports of the Commission on Engineering Education and ASEE present a clear challenge to both educators and computer manufacturers. They indicate the need for training in analog and hybrid computation for all engineering disciplines.

This paper will briefly address itself to three questions:

1. Why are these recommendations important?
2. How are engineering schools implementing these recommendations and what other action can be taken?
3. What analog and hybrid hardware and support are available to the educator who is implementing the recommendations?

HYBRID COMPUTERS

Analog and digital computers made it possible to solve problems that would have been impossible or economically impractical before their advent. Indeed, they have stimulated new thinking and have made it feasible to define new problems in order to push forward research frontiers. Many exceptional achievements in the life sciences, the process industry, and the automotive and aerospace fields are the direct result of using the powerful tools of digital and analog computation.

Hybrid computers have grown out of a need to analyze and solve *new* problems which cannot be solved economically by purely digital or analog techniques. They combine the high speed of the analog computer and the program flexibility and precision of digital logic and the digital computer.

The simplest application of a hybrid computer is the simulation of a system which is naturally hybrid. Examples of such systems include: attitude control of a space vehicle, certain functions of the nervous system, and direct digital control of a chemical process.

Optimization problems requiring automated, high-speed, iterative solutions are particularly suited to hybrid computers. Dynamic models are simulated on the analog computer, and evaluated and modified on the digital computer. Problems in this category often demand hundreds of thousands of integrations of differential-equation sets. Only a hybrid computer can do this within a time which will make the design economically feasible.

The solution of multi-dimensional, partial-differential equations is also only practical when analog (parallel) integration supplements the digital computer. Consider a particular example, that of a heat exchanger involving three states (liquid, vapor and heated vapor) with variable boundaries. A power company evaluated programming and debugging time at six months if the system were simulated on a large digital computer. To arrive at an acceptable design, over 100 solution runs would be necessary. Prior calculations showed that each digital run would require up to two hours. On a hybrid system it took less than a month to program and debug the problem, and only two days to arrive at a satisfactory design.

There are already over 150 hybrid installations and the rate at which hybrid systems are being purchased is at an all-time high. These hybrid facilities have grown from a need to simulate and study systems which could not be simulated economically on either digital or analog computers, although most companies with hybrid capability also have separate digital and analog facilities. In these companies the design engineers can choose the most appropriate "tool" to solve their particular problems.

In education, hybrid computers and computation concepts are in their infancy. The interest on the part of educators, like yourselves, has been intense, because of the recognition of the excellent potential of the hybrid computer as a tool for engineering design. A number of schools already have taken the initiative and acquired hybrid-computing capability. These include schools like the University of Minnesota, Worcester Polytechnic Institute, the University of Texas, Harvard, the University of Michigan, Stanford, UCLA, and the University of Wisconsin, just to name a few.

These schools and others have recognized that problems demanding hybrid computation are not in the future, but are being defined and must be solved today. Industry must have engineers who are capable of using these systems, and today's students and practicing engineers must be taught hybrid concepts in undergraduate and continuing education programs. Recent workshops sponsored by universities and industry have been filled to capacity, a testimonial to the need for continuing education in hybrid simulation. The design engineer who does not appreciate this type of simulation is severely limited in his conception of problems. He runs the risk of solving problems in increasingly inefficient ways, or not at all.

To date, courses in hybrid computation have been confined primarily to graduate work. A small number of courses have been opened to upper-division undergraduate students, but there is a need for more than a slight increase in the trend and the number of these programs. The ASEE report calls for a lower-division course to acquaint students of all engineering disciplines with hybrid computation; this recommendation underlines the importance of early action by educators and industry.

ANALOG COMPUTERS

The substantial number of analog computers already in use in classrooms testify to their value as teaching tools. In the past few years, EAI alone has supplied more than 600 machines to colleges and universities.

The computers serve as mobile classroom demonstrators which permit teachers to demonstrate dynamically solutions of complex, non-linear differential equations. Analog computer simulation makes it possible for the student to appreciate the values of described physical systems mathematically. It can, and does, penetrate the barrier which the average engineering student finds when he tries to relate mathematics to the real world.

The analog computer answers the student's questions about how the physical system behaves under different conditions. Instead of equations that are impossible to solve, he sees a real solution. More important, his questions about changes in parameter and initial condition values are answered immediately. The instructor, or student, simply turns a potentiometer or throws a function switch, and sees an immediate response. He has an excellent feeling for the system because there is a one-to-one relationship between the physical system, the mathematics, and the computer model.

Instructors are constantly relating stories about the enthusiasm and interest of students who are taught with the aid of analog computers. The poorer student is able to absorb basic concepts which are difficult for him without the aid of the computer, while the bright student can get answers to much more complex questions.

A significant point about the use of the analog computer as a lecture aid is the fact that the student does not have to know how to program the computer. A short lecture supplies all that is needed to relate the computer solution to mathematics and the real world. The implications for teaching differential equations as early as the freshman year are particularly appealing. By developing an early appreciation of mathematics, the student is ready for exposure to design courses early in his academic training.

The activity of the Engineering Concepts Curriculum Project of the Commission on Engineering Education has actually introduced analog computers at the high school level. The computers are used to give the student an appreciation of dynamic systems and the concepts of engineering design.

In the laboratory, the course in analog computation often gives the engineering student his first experience with the design process. After acquiring the fundamentals of computer programming, the student is usually given the opportunity to choose one or more design projects involving synthesis, programming and optimization through experimentation on the computer. If it is important that we teach engineering design, then I suggest that the analog computer has been, and will continue to be, one of the most powerful teaching tools.

The analog computer with its peripheral equipment is also an excellent tool for research. The "hands-on" use of the computer with the man in the loop provides incomparable insight into the operation of a physical system. It is an economical form of experimentation offering the flexibility of continuous adjustment of parameters with immediate display of new solutions. Programs can be changed directly on line by repatching all or parts of the program on the computer.

The analog computer is often used to examine problems that are poorly defined, that is, to develop insight into physical systems that the designer does not really understand and cannot describe mathematically. Through intuition and "gut feel," the designer begins to guess at the mathematics underlying the physical system. He may construct several subsystems on the analog computer and attempt different types and amounts of cross-coupling until his solutions begin to resemble the behavior, or predicted behavior, of the physical system. When his results agree, he can go back and determine the mathematical description. The next step might be a larger, more complex analog analysis, or the use of a hybrid or digital computer to refine his analysis.

The strength of the analog computer as a teaching tool for the student of engineering design should be apparent. It is economically feasible to have four or five, or more, consoles in the laboratory, particularly with EAI's liberal educational contribution program. Small groups of students, or individual students, can work on a design project with maximum man-machine intimacy through the power of CRT displays, XY and strip-chart recorders, and digital voltmeters. The students' curiosity, drive, creativity, and desire to investigate further are inspired by the immediacy of hands-on parameter adjustment and instant readout of complete sets of solutions.

The analog computer not only offers the student the opportunity for involvement and participation in engineering design, but also teaches him disciplines of analysis which are very valuable in constructing and conceiving realistic models. In addition, he acquires basic concepts which will be necessary in teaching him the fundamentals of hybrid computation.

The list below shows the departments already using analog computers in design education in engineering:

Aeronautics and Astronautics
Agriculture
Chemical
Civil
Computer Science
Electrical/Electronic
Mechanical
Nuclear

This list clearly indicates the need for a basic, sophomore, introductory course in analog simulation, as recommended by the ASEE subcommittee on computers in engineering education. This approach would certainly be more efficient than programs which require basic analog-programming courses within all the different departments. The basic course opens doors for educators in all departments, not only in the student's major, but also in mathematics, physics, and chemistry. If all of the students receive basic training, the instructor can employ the analog computer whenever it can best fulfill a teaching requirement in the classroom or laboratory.

ANALOG/HYBRID HARDWARE AND SUPPORT

Modern analog and hybrid hardware for educational applications employs solid-state design and does not require special environmental control. The systems have been designed for low maintenance and "student-proof" operation. They employ modular construction to permit maintenance of modules without disrupting system operation.

EAI offers a complete line of analog and hybrid computing systems to serve both education and industry. The TR-20, and its predecessor, the EAI PACE TR-10, are the most popular analog computers for educational use. They are used extensively in teaching basic analog computation to undergraduates in engineering and science. They are also used heavily in lecture demonstrations and in both undergraduate and graduate laboratory research and design projects. A special educational contribution policy makes the TR-20 particularly desirable for schools.

The EAI TR-48/DES-30 Analog/Hybrid Computing System is the standard for both industry and education in desk-top class equipment. More than 500 of these systems are in operation in classrooms and laboratories, serving principally as tools to aid in engineering design and research.

The EAI 680 Analog/Hybrid Computing System is the latest addition to EAI's product line. It has been designated for tie-in to a general-purpose digital computer, and offers powerful hybrid computing capability at a price within the university budget.

The EAI 8900 Hybrid Computing System combines the EAI 8800 Analog/Hybrid Computer with the EAI 8400 Digital Computer. This system is designed to satisfy the needs of large simulation laboratories in government and industrial installations. It is the type of equipment, along with the TR-48/DES-30 and EAI 680, that today's student will be using to solve engineering design problems after he graduates.

In addition to hardware, catalogs of courses are available to educators, as well as to industry. These educational programs start with basic programming and extend to sophisticated computed application concepts in most disciplines. EAI offers a library of laboratory problems for student work in most engineering-design areas. This reference library has a number of pre-programmed demonstration problems as well. Programmed student workshop material, slides, films, and other teaching aids have also been developed by educators for classroom use.

SUMMARY

Many people are using analog and hybrid computers in engineering design. The use of these tools is growing rapidly as new simulation areas are discovered. The design engineer who cannot appreciate and use analog and hybrid computers risks solving problems inefficiently in time and cost, if, indeed, he can solve them at all.

Engineering educators in all disciplines use analog and hybrid computers as aids in teaching engineering design, as well as in teaching students the concepts necessary to use these tools in industry. These computers serve equally well as mobile classroom demonstrators, student laboratory instruments and as effective tools for design and research. In line with the ASEE subcommittee recommendations, the universal value of analog and hybrid computers in engineering design strongly suggests the need for including analog and hybrid concepts in a lower-division undergraduate course.

Economical, modern solid-state hardware, together with available educational software, make analog and hybrid computers particularly appropriate for educational use.

RECENT DEVELOPMENTS IN ON-LINE, REAL-TIME TIME-SHARING AND MULTIPROGRAMMING COMPUTER CAPABILITIES

A. M. ROSENBERG
Scientific Data Systems
Santa Monica, Calif.

Computer time-sharing and on-line use, basic tools for education in engineering design, are areas in which we at Scientific Data Systems, and myself in particular, are focusing special attention.

On-line use doesn't necessarily involve time-sharing; in the early days of computer development, whenever a person went up to the computer and used it, he was on line, and he could do whatever he wished with the machine. But as computers became larger, faster, and more expensive, this procedure became increasingly impractical; and time-sharing evolved naturally from practical and economic considerations.

One definition of time-sharing is the following: a number of users share a computer in an interactive environment. At SDS we take into consideration the entire spectrum of time-sharing, which includes critical real-time operation (such as process control), interactive conversational use, and background production (a job-stack operation).

There's nothing wrong with having a complete mix of all three, particularly if they are related; it's only a question of scheduling and response time.

Our new product line, the Sigma series, is specifically designed to process simultaneously business data, solve scientific problems, perform real-time control, and interact with multiple-user stations.

First, I'd like to review what SDS has done in the past with regard to time-sharing or on-line use, particularly in the area of design. SDS has one product, the Differential Equation Solver (DES-1), which is based on a general-purpose digital computer but which has a special console for on-line manipulation of problem parameters and timing cycles, giving the user a combination of capabilities.

The language is based on FORTRAN, and either the console typewriter or the card reader creates the operating program. The analog inputs are controlled by the operator at the DES-1 console. This console permits on-line monitoring and control of problems during execution, along with strip-chart recorders, scopes and plotters.

The flaw in this approach (one-man use of a general-purpose machine) was that the user was keeping the entire machine tied up, although he was not actually using it all the time. An obvious course was to share and interact with the computer, still giving the DES-1 user priority because once he had started problem execution, the time cycle became very demanding and critical. The problem was that there was only one channel available for the typewriter, and a second user would have to use the same typewriter. If the second user had had another I/O (input/output) channel, he would have been able to time-share the machine, because any on-line use of a machine involving a human operator involves delays. If all of the core memory is not being used, the other user can take up available CPU time. With the DES-1, the user had conversational use of his simulation program because he had on-line editing capability. In addition, he had on-line intervention by means of the DES-1 console.

Most of our machines at SDS are being used in on-line scientific applications, and, more specifically, in real-time data acquisition, telemetry work, nuclear work, biomedicine, etc.

One of our creative customers, the University of California at Berkeley, purchased an SDS 930 computer and modified it for multiprogramming. This modification allowed dynamic relocations, memory protection, and a number of features that would, for example, allow the use of common public routines. This machine is available from SDS as the 940 computer.

All aspects of time-sharing are featured in the SDS 940 including multiprogramming; real-time processing; on-line, remote data processing; and simultaneous access to the central computer by several users. The SDS 940 time-sharing system provides up to 128 users with simultaneous access to the problem-solving capabilities of the high-speed digital computer. A response time of two or three seconds facilitates rapid communication between the user and the 940.

So far we've discussed hardware but in the time-sharing business, it's the software that counts. Berkeley developed outstanding software and it's possible to see it, as well as to use it, in operation today. The Berkeley system has been in continuous use since April, 1965, and demonstrations of its ability to service multiple, remotely located users have been made by linking the computer with teletype consoles at the University of Illinois, Urbana, Illinois, and at Stanford Research Institute, Palo Alto, California. It is a general-purpose, time-sharing s stem in the sense that there is full memory protection, and the user can work from the assembly language level on up. As a result, they've built many sophisticated tools, and have done this easily because of a set of common routines, involving text-handling, etc., which provide building blocks for other applications.

The Berkeley time-sharing system has a macro-assembler, a very sophisticated debugging package, a sophisticated editing package, LISP, SNOBOL, FORTRAN, and ALGOL, in addition to Auto-Secretary, which is a documentation editor. They were able to create these and many other programs very quickly with the on-line, time-shared facility.

A major feature of the Berkeley system, from the point of view of teaching and ease of use, is full-duplex communication. We all want to have our cake and eat it too. The user would like to talk to the machine, but perhaps he is not a good typist, or perhaps he is lazy. Yet he wants to see his output, and he'd like to see it in English rather than in hard-to-read shorthand. The full-duplex communication allows every character to be processed and turned around, and that processing can actually change the input character in dramatic ways.

With the Berkeley system, any kind of executive-system or subsystem command can be typed in; when the system recognizes what the user is really trying to say, it will finish it for him—it's almost alive. If the user doesn't require this help, he can tell the system he's an expert rather than a novice, and it will react by merely accepting the shorthand notations without completing the full text.

Every on-line system and every on-line application is a potential learning mechanism because when the user is interacting, he is learning, even when he makes mistakes. There is a teaching mechanism called HELP that has been built into the Berkeley system as software. It takes natural English text in a question-answer form and looks for key words. It also recognizes the question in the context of what the user is asking about. For example, if he is asking about string manipulation for SNOBOL, a programming language especially suited to operations on strings of characters, HELP knows (in context of SNOBOL rules) what the user is asking and will give back the appropriate answers. Every subsystem, every service system, and every language system can also be connected to HELP so that the user can always have HELP either directly, or while he's working with some language or process.

The Berkeley system for the SDS 940 capitalizes on the tasking mechanism similar to Programming Language 1 (PL/1) or PL/2, which they call a FORK. By means of the FORK facility, programs can call other programs, jump back and forth from one program to another, and take any kind of program (for example, one that is a design tool) and put it under the control of a teaching mechanism such as a programmed instruction package. To gain experience, the programming tool can also provide the mechanism for guidance—actually leading the student by the hand.

At SDS our manufacturing personnel are running circuit-design problems on-line using a typewriter as a communications tool. They are also using a logic design program that is based on a version of FORTRAN to which have been added logical operators such as: AND, OR, INVERT, BUFFER, and so on. These programs allow labels to be much longer than 26 characters, and the user can actually insert a logic equation as a label.

The next important step for the future is packaging, which involves putting the modules together compactly so there are no timing problems caused by separation of the modules.

Documentation information is stored on magnetic tape and used in testing the final product for customer service needs, and any other documentation to support a product we develop. In this manner SDS is taking advantage of the advanced techniques which evolve from the development of our own products.

A new item on the SDS marketing list is our third-generation computer, Sigma 7, which has been designed for the full spectrum of time-sharing. For real-time applications, Sigma 7 hardware capabilities feature high-speed computation to keep pace with real-time processing demands; flexible input/output to permit handling a variety of types and sizes of data easily; almost instantaneous response time with an improved priority-interrupt system that always reacts within microseconds; and the ability to change the entire state of the machine from normal processing to interrupt processing, while saving all necessary status information and interim results. A single instruction requiring only six microseconds provides this context-saving capability. In addition, high reliability is achieved through use of SDS-designed, monolithic, integrated circuits and special production and automatic checkout equipment.

Field-proven peripheral equipment for use with Sigma 7 includes keyboard printers, card readers, card punches, line printers, magnetic-tape units, rapid-access data files, paper-tape readers and punches, graph plotters, display equipment, and data communications equipment. SDS graphic scopes will be particularly useful in on-line design applications.

We feel that the system should be centralized because of a relationship between what the user may be doing, for example, and a real-time process, a conversational process, or perhaps a secondary background process. There is no single job that is necessarily restricted to only one mode of operation, and the jobs in various modes may be related.

For example, suppose the user has a real-time control process and wants to improve it or wants to change some parameters, developing the change first and debugging it before actually controlling something. The mechanism allowed in the hardware and software can break up such a process into, first, the very critical, responsible, sensitive part dealing directly with the input/output; and, secondly, the part which is the normal computation or processing program, which could be full of errors and ought to be caught and trapped if it is making a mistake.

At SDS we do this by the programmed operator mechanism. In the Sigma series, just as in the SDS 940 computer, there are programmed operators that are public routines operating trap locations. These operate in the master mode, and any machine operation can be legally performed. Some of these trap locations can be dedicated or allocated to a real-time process by saying, "Please load me in." The user can specify what part is sensitive (for example what part should operate directly with I/O and be plugged into interrupt lines) and what part is a normal kind of process to be treated as a slave or user program. The latter, of course, can be kept in core memory and not swapped out, while normal jobs may be swapped out because of multiprogramming activity.

The Sigma 7 has selective memory protection. It has dynamic relocatability through a map which also allows fragmentation of programs. The pieces of a user's program can be moved around anywhere in physical core memory. Input/output buffers can be left in core while the rest of the program can be removed at will. Through the software, tasking facilities are provided, which means that an entire program need not reside in core unnecessarily.

Sigma 7 can also control up to eight input-output processors in various combinations of two types. Each multiplexor I/O processor can accommodate up to 32 I/O channels operating simultaneously. Each selector I/O processor can accommodate up to 32 very high-speed devices, only one of which can be operating at any given instant. Combining these two types of processors permits the user to configure the exact combination of simultaneity and speed required. The I/O processors operate semi-independently of the central processor, leaving it free to provide faster response.

SDS Sigma 7 software offers a totally integrated and comprehensive hardware/software package and includes FORTRAN IV and PL/1 compiler in both standard and high-efficiency versions, because SDS recognizes that hardware and software have become so completely interdependent that comprehensive software is now essential to full-system utilization.

The difference between conversational and batch processing makes little difference because we use essentially the same compiler. We provide on-line editing similar to the Dartmouth system with one exception: with Sigma we provide for closer user interaction. We do not wait until the user finishes typing in all his statements to have the compiler look at them and say, "You have made the following mistakes." There is a connection between the editor and the syntax analyzer of the user's compiler so that, as the user enters each text line, he is told immediately what is grammatically wrong. This provides a step-by-step analysis and reporting of errors. The program can then be compiled and executed and the input cycle repeated if the user finds logical program errors. This interactive relationship will be available with FORTRAN and PL/1, and the mechanism is general enough so that any language that we decide to add to the software system (such as LISP II or COBOL) can be handled by the same mechanism by merely plugging the syntax-analyzing portion of the language processor into the on-line editing package. The on-line editor is essentially what makes the languages conversational.

With Sigma 7 we are also planning to allow the DES-1 concept to be expanded, but in a time-sharing environment, because it is not efficient to tie up an entire machine unnecessarily. Because of time constraints and scheduling, it may be possible to have many on-line users getting one- or two-second responses at the same time a DES-1 user is operating, but batch jobs can be run in the background. This is why we say that the whole spectrum can be accommodated; the facilities are there and the user can operate a Sigma 7 system to accommodate his needs.

A frequently asked question is, "How many users can you put on your time-sharing system?" The answer is, "How many users doing what, with what, and requiring what kind of response time?"

If it's a real-time process that has certain demands on the CPU and core storage, some capacity will necessarily be lost. We are generalizing our scheduling mechanism so that the number of priority queues (that is, the rules for operating in each queue: how much of a slice of time should be given, does it run to completion, does it take over the whole machine, etc.) are the parameters that the user will have to determine in terms of his own installation and applications.

At this time users haven't clearly defined their own needs. Some say that a response time of ten seconds is sufficient. The Berkeley designers, for example, won't tolerate anything over one second for a conversational process; whereas, in the System Development Corporation system, the range, which was constantly changing, was anywhere from one second to approximately 15 seconds. It is my view that consistency can be even more important than any particular set of times, and every user will have to determine for himself whether or not he wants gradual degradation. To prevent gradual degradation of response time, a cutoff point (indicated by a busy signal) can be established so that no new users would be accepted by the system.

The teaching of engineering design will be greatly enhanced by recent developments in on-line, real-time, time-sharing, and multiprogramming, computer capabilities—dimensions keyed to the needs of dynamic change. These needs have been reflected in the design of the SDS Sigma series computers. With the modular Sigma concept, system configurations can be expanded to fit the needs of virtually every operating environment for a variety of creative applications.

GENERAL ELECTRIC REACTIVE DISPLAY SYSTEM

MARVIN T. S. LING
Graphic Techniques Project
General Electric Computer Equipment Department
Phoenix, Ariz.

GENERAL ELECTRIC REACTIVE DISPLAY SYSTEM

INTRODUCTION

Perhaps one of the most important developments connected with the computing field during the last few years is time-sharing. By this we mean that a number of users share the same computer simultaneously and the response from the computer is fast enough to satisfy the needs of all users. Essentially, it implies that every user of the computer has direct access to the system. The development of time-sharing is very important, since it offers at least the following advantages:

1. It eliminates the inefficiency caused by the present red tape programming of a hierarchy of programmers.
2. Because of fast turn-around time, it improves human productivity.
3. It increases the consumption of computer power by an increase in the number of problems submitted to machine solution which were never previously submitted because of the inconvenience of using the computer.
4. It provides the opportunity to work with a class of problems in which the interaction between computer and user may facilitate or lead to a more creative solution.

Focused on human direct access and the on-line use of computers, the following matrix of techniques for man-machine interaction is provided:

A. Keyboard
   Media—punched cards, alphanumeric texts in hard copy
   Machinery—keypunch, printer, Teletype equipment
B. Visual/Keyboard
   Media—A + alphanumeric texts in display
   Machinery—A + display tubes
C. Control Wand/Visual/Keyboard
   Media—A + B + pictures
   Machinery—A + B + control wand
D. Voice/Control Wand/Visual/Keyboard
   Media—A + B + C + microphones
   Machinery—A + B + C + microphones and speakers

The General Electric Reactive Display (GERD) system is an experimental system focused on the use of technique C above. By using a hand-manipulated control wand and display scopes, the user communicates with the computer by means of textual as well as pictorial languages.

OBJECTIVES

The GERD system has two major objectives:

1. To expand and maximize the total user value derived from present and future computer systems. To achieve this, GERD should provide many remote display terminals time-shared with a central computer system so that the user accessibility can be increased. The communication languages used at each display terminal should be both textual and graphic. Graphic communication is important in that human beings understand graphics quickly and the presentation is concise. Furthermore, each terminal and the system should be designed in such a way that users of various backgrounds may use the system on-line with a minimum of learning required.

2. To ensure that the basic GERD software packages be general, flexible, efficient and open-ended, GERD should provide a horizontal software package which allows users to generate their own specific application programs within broad compatibility boundaries. The total number of applications users may make at the display terminals is practically unlimited. GERD initially categorizes those applications in two areas: the engineering and manufacturing applications, namely, design and numerical control; and the information storage and retrieval applications for management.

SYSTEM DESCRIPTION

To achieve a system such as that described above, proper organization of information within the computer is of critical importance. A general data structure should be developed for efficient retrieval, updating, and storage. In particular, it should be able to make associations easily between stored information and new ideas. With this kind of general data structure, it is possible to operate every application problem on the common data base by common primitives and subroutines. For example, the information as to how many bolts are used on part A, or how many cars are red, will be stored in the same general data structure as the information for what line and curves should be machined next by a numerically controlled machine tool.

GERD should be designed for users of various backgrounds. Mechanical and electrical engineers may use it for their various design problems. The manufacturing engineer may use it for 2-D or 3-D numerical control of parts production. Or he may make use of an X - Y plotter located near the console for more precise drawings after the desired part has been sketched at the console. Draftsmen may use the system to generate new drawings and store them, as well as to obtain hard copy. Or they may make use of the image-processing unit to bring an existing blueprint into the computer and analyze or modify the drawing at the display console. Engineering drawings may eventually be stored in the mass storage media, with hard copies obtainable remotely through the terminals. When GERD is to be an on-line management or command-and-control system, the users should have complete control over the system performance. At his option, the user may select various data formats for display. He may analyze the displayed data and up-date it at the console.

Since the total amount of information that can be displayed at one time on a display console is limited, GERD should allow an initial display of the outlines of entities and a further display of the detail of a particular entity selected by the control wand. The display scope should be treated as a window that can look either at any particular portion of information, or at more information with less detail. GERD allows the display of perspective views of three-dimensional objects. Rotations of the displayed object on three axes can be done easily by the users.

The graphic language to be made available on GERD should include basic functions such as drawing lines, circular arcs and mathematically described freehand curves; it should also provide constraint capabilities (making one line parallel or perpendicular to other lines, tangent to a curve, or of a certain length, etc.) and such functions as erasing, and moving, etc. It should also be capable of generating mathematically described three-dimensional surfaces, given some boundary conditions. The users can easily make modification of surfaces through a control wand or through pushbuttons. The users of GERD may have the choice of eliminating hidden lines or having them remain displayed. Since the hidden-line elimination program is rather complicated, the system should not impose extra computer time on the users if they can accept the hidden lines.

In addition to graphic languages and languages for computer-aided design and for information storage, retrieval, and control, GERD should provide logical and arithmetical operators so that users may name and execute procedures, and then store and retrieve them. This facility can be constructed out of all existing languages.

Finally, GERD should be easy to use. No special training should be required. Users should not be required to have any knowledge of computers, although this may help them to make more creative use of the system. The design of the system should take human-factors engineering into consideration. The number of pushbuttons should be limited, so that users will not have to spend time searching for particular buttons. Consequently, a large number of functions should be displayed, with a particular function being selected by the control wand. The languages should have a syntax and vocabulary complex enough to permit a flexible range of alternatives. To assist communication between the user and the system, GERD should continuously display the next level of languages in inquiry form. That is, the system should continuously assist the users in forming the expressions they wish to communicate. Furthermore, the user should not be required to keep track of the syntax he has formed at any time; and he should not be worried as to syntax restrictions and other possible constraints.

GENERAL DATA STRUCTURE

The importance of having a very general data structure in any powerful system can never be overemphasized. In order that GERD be general and flexible, the display terminals should be able to handle all possible kinds of entities. An entity is defined as anything in this real world which has certain properties. These properties can be separated into three categories. First, an entity has some properties which are unique to itself. A triangle may have been drawn by the designer to represent a support of a bridge with infinitive rigidity. This information, as well as the date, project number, and name of the designer, are data of the entity not common to others. Second, an entity may contain other things in various ways. If lines and points are regarded as entities, then a triangle contains three line- and three point-entities. Finally, an entity may very well be contained in some other things. From the previous example, the line is an entity; and it is contained in other entities—namely, points and triangles. Consequently, an entity always associates with other entities, if one wishes to define them as atomic levels. It is thus very important that the general data structure should be designed in such a way that entering properties of things and making associations between things can be easily accomplished. The general data structure used by GERD is shown in Figure 1.

The square blocks represent entities, while circles represent "conjunctions" which contain the description of the relationship between two related entities (the usage description). The mechanism used to link entities and conjunctions is a "ring" structure, although other types of structure could

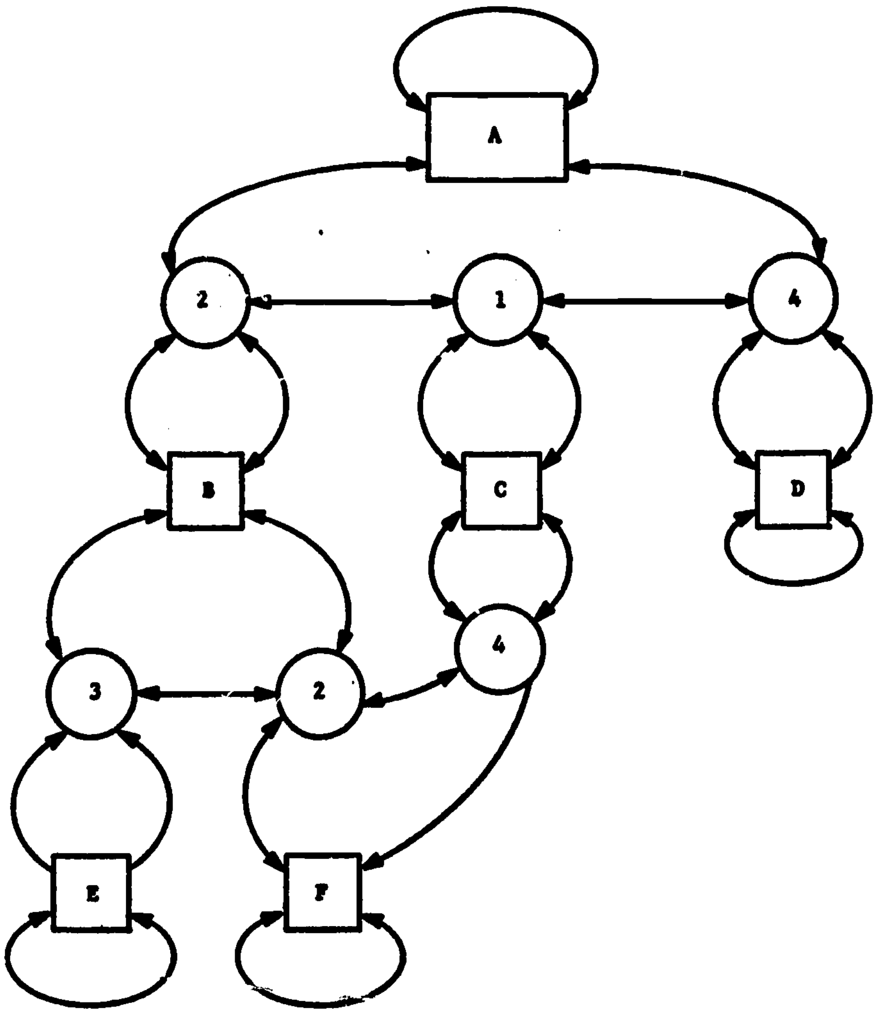


FIGURE 1 - EXAMPLE OF A GENERAL DATA STRUCTURE

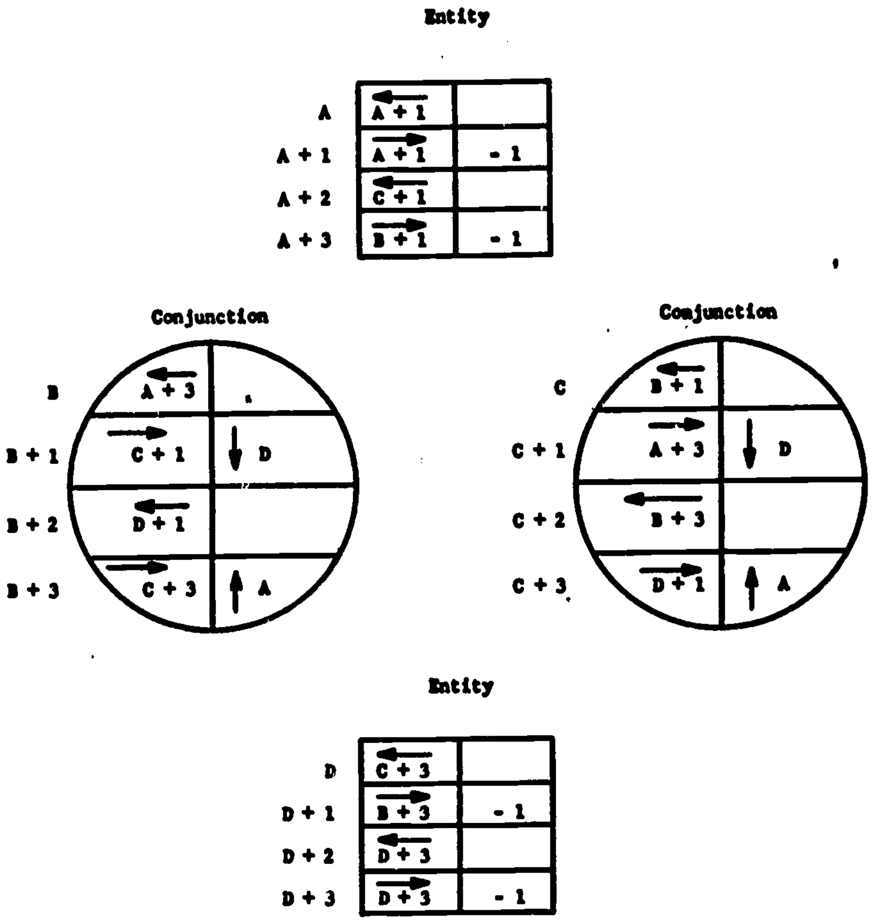


FIGURE 2 - RING STRUCTURE EXAMPLE

also be used. A ring is a special form of list tie. Each element in a ring consists of forward, backward, and head pointers. Forward and backward pointers close themselves and form a ring.

Basically, any entity will have two types of rings, those that relate entities it contains and those that relate entities in which it is contained. We call these two types "call-out" and "where-used" rings, respectively. The head pointer of a call-out ring is then pointed to the head element of its where-used ring, and the head pointer of a where-used ring is pointed to the head element of the call-out ring (for our data structure, we would never search for the head element of the same ring). The ring structure is shown in Figure 2. The unique properties of an entity are stored in the entity block, or its extension. The example shown in Figure 2 indicates that an entity A consists of 2 entities B, 1 entity C, 4 entities D. An entity B consists of 3 entities E, 2 entities F, and so forth.

LANGUAGES

On an experimental basis, part of the languages outlined here have already been implemented for the GE-435. It is not our present intention to cover all possible languages which all future users would like to see available at the console. Languages for a particular application, such as tool selection, spindle speed, coolant on/off, etc., in numerical control, are not included.

Since the number of required languages is already large, and will grow as more and more application programs are built in, it is important that the console should be designed in such a way that the user will not be required to search for the required function keys. Basically there are two ways to communicate the desired languages to the computer: by function keys or by function pads. Function keys are pushbuttons, each assigned a particular function by the system designer. Function pads replace function keys by displaying symbols representing functions in the oscilloscope, and the selection of each function is made by the control wand.

GERD will have both function keys and function pads. The total number of function keys will be rather limited. The use of languages will be in the conversational mode, and they will be in hierarchal order.

The following are brief overall descriptions of languages designed for GERD. Each of these languages represents a group of finer languages.

1. CREATE—This allows the user to create new entities. These entities may take the form of two-dimensional or three-dimensional pictures or surfaces. Textual information is considered two-dimensional. The user may indicate either outlines or details of entities, and may give them symbolic names. It also allows him to create new display formats.

2. STORE—This allows the user to store the information (entities or data) into mass memory. The operator may be requested to indicate if the

entity should be stored in the where-used or call-out rings of other entities.

3. DISPLAY—It allows the user to display three types of information:

a. *Previously stored entities.* If entities are not in core memory, the system will bring them into it from the mass memory. The user may specify if outlines or details of entities are to be displayed.

b. *All constraints applied to the current pictures.* This is to enable the operator to erase some of the constraints and further assist maze-solving for a constraint-satisfaction program.

c. *Information satisfying given conditions.* This is to allow display of all information, satisfying the given conditions expressed in Boolean algebra form.

4. DRAW—This allows the user to draw points, lines, circular arcs, freehand curves, and surfaces, on the display scope. Points can be defined by control wand alone or by keying in X, Y with reference to a given point. If the control wand is used, any point on the display can be selected from those points physically located on existing things, such as the intersection of two lines. A circular arc can be defined either by three points on the arc, or by the center, starting, and ending points of the arc. A fillet—that is, a circular arc tangent to two existing lines—is also definable. Freehand curves are mathematically defined. Surfaces may be defined either by sketching boundaries or keying-in boundary conditions.

5. REPRODUCE—This allows the user to call upon previously sketched entities either in the form of copy or instance. If it is copy, the internal data structure is accessible by the control wand; and the data structure of the copied entity is merged with that of the current picture. If it is instance, the entire entity is treated as one item; and access to its internal data structure is prohibited. REPRODUCE can be in its original form or in mirrored form with respect to its origin, the X-axis, or the Y-axis.

6. MOVE—This allows the user to move appointed things either in the association or the disassociation mode. In the association mode, when a "thing" is moved, all others depending on that ring are moved correspondingly. On the other hand, in the disassociation mode, only the thing pointed at by the control wand is moved, and is made independent of all other things.

7. MODIFY—This allows the user to modify the things displayed. The modification may be resegmentation of lines or reshaping of freehand curves or surfaces.

8. ERASE—This allows the user to erase the entire screen, an individual thing, or an entity.

9. WINDOW—This allows the user to view the objects in the following various ways:

a. To rotate the picture around a specified axis.

b. To scale either the specified individual picture or an entire screen.

c. To turn the pages vertically or horizontally, primarily for textual information in table form.

d. To select and display a particular view of a three-dimensional object.

10. CONSTRAINTS—This allows the user to apply geometrical constraints to the drawings, inasmuch as the inaccuracy of the display scope, plus human errors, prohibits creation of accurate drawings. The constraints include:

a. Vertical
b. Horizontal
c. Perpendicular
d. Parallel
e. Merge
f. Fix
g. Tangent
h. Length
i. Angle

11. VALUE—This allows the user to specify the nature and ranges of ordinate and abscissa in Cartesian coordinate systems and further allows them to request the display of a characteristic curve of some system. It also displays data of particular points on the curve and the value of things, which include the dimension of a line and an angle.

12. MATH OPERATORS—This provides the user with a set of mathematical operators. These operators include *, +, /, -, Σ, NEGATE, INVERSE, ABSOLUTE.

13. LOGICAL OPERATORS—This provides the user with the following set of logical operations: LOAD, STORE, >, <, ≥, ≤, =, , AND, OR.

14. MATH FUNCTIONS—This provides the user with the following set of functions: MAX, MIN, MEAN, SQUARE, SQRT, SINO, LOGE, EXP, ARCTANO.

15. SYSTEM—This allows the user to start and stop the display terminals, dump the data structure, obtain a hard copy from the precision plotter and convert the information read into the data structure by the image-processing unit.

16. MISCELLANEOUS—This allows the user to request removal of hidden lines, hatch the specified area, specify the scale of things, designate attachers, and undo the last function.

REFERENCE

John W. Weil, "The Impact of Time-Sharing on Data Processing Management," Data Processing, Vol. 9, 1965.

ACKNOWLEDGMENT

I would like to express my deep appreciation for the encouragement and support given to this project by the General Electric Computer Equipment Department; especially, to Dr. Leonard Maier, Dr. John Weil, John Couleur, Larry Hittel, Linc Young, and Frank Yao. Many of my colleagues have contributed ideas in developing the GERD system. The concept of reversing the head pointer in ring structure was originated by Mrs. Sandra Palais.

CONTROL DATA'S ACTIVITIES
Basic Hardware and Software
Peripheral Equipment
Application Program

W. Beilfuss
Meiscon Corporation
Subsidiary of Control Data Corporation
Chicago, Illinois

With the expanded product lines of today's computer manufacturers, it becomes difficult to describe briefly all of the products that will affect engineering design. These product lines include a range of computers which vary widely in both size and function; a growing list of peripheral devices for different forms of input and output as well as for data storage; and a broadening software and application program library. To cover adequately all of Control Data's activities and their implication would take more than the time allotted. I will try to touch only briefly upon those aspects of our activities that seem to be most widely known, and to dwell somewhat longer on those less widely publicized, especially as they are concerned with graphic processing and engineering program development.

BASIC HARDWARE AND SOFTWARE ACTIVITIES

The goals to be achieved are reduced cost of computation per job, reduced turnaround time, enlarged problem-solving capabilities, and greater capacity to systematize the man-machine loop. To achieve these goals we are developing faster and more efficient CPU's enlarged parallel-processing units (such as the 6600 which has ten separate processing units, all of which share an eleventh), and a greater use of time-sharing. Problems which were beyond the capacity or economical handling of previous computers are currently being attacked on 6600's. Worldwide weather forecasting, in which the earth is being thermodynamically modeled in increasingly greater degree, illustrates this enlargement of problem capacity. Probably the computer advance which currently generates greater interest is time-sharing, for this concept attacks the problems of computational cost, turnaround time and machine accessibility.

Control Data's approach to time-sharing includes the following capabilities:

1. Multi-processing—automatic operation of computer systems with two or more programs of different priority being submitted in batch mode. Operation is shifted from program to program based upon changing priority.

2. Multi-programming—two or more programs active in central memory. Using various internal and external criteria, control is transferred rapidly from one program to another.

3. Multi-access—many devices having access to the system. Terminals include customary peripherals; unit process control devices; or people in a remote, on-line, real-time basis.

Each of these activities require high-speed internal processing which can significantly outperform the I/O times, interrupt capacity to break program processing when an external signal calls for service, and an address transfer scheme to relocate programs being processed when they must be transferred in memory. Control Data currently approaches time-sharing with two offerings: the 3300 and 3500 computers, and the 6400 and 6600 computers.

In each of these systems, the whole range of time-shared problems can be intermixed and handled through a combination of hardware and software. The 3300-3500 computers use interrupts, registers and software to process the programs in small segments called "pages." The 6000 series, with their extended memories and faster transfer rates, utilize a "memory swap" arrangement to exchange core in large chunks. Regardless of the technique, the engineer's request for computation from his remote console will be honored automatically, with due priority. His requests will be handled along with all other programs that are in line for processing, just as if the engineer were on-site. The cost, due to efficient sharing of the machine with others, will be based solely upon computation time, and that time is much reduced.

In support of its hardware, Control Data currently provides complete operating and extended compiling software, including that necessary for time-sharing operation. In addition to these standard items, software such as APT, process control software for the 1700 computer, and certain special software, such as management "Request Language," is being developed. The latter is an English language command-set to "call-out" management data from a company central business file and display it on either typewriter or CRT console.

PERIPHERAL EQUIPMENT

The peripheral equipment that provides the greattest assistance to engineering problem solvers can be classified in two general categories: mass storage, and input/output. In the former, Control Data provides large disks of up to 200 million bits of storage, 4 million character drums with 2 million characters-per-second transfer rates and the familiar disk packs. Continued advances in this area are planned. However, it is in the area of graphic I/O devices where new ground is being broken for design engineers. Document readers, high-speed CRT graphic hard-copy devices, film scanners and conversational mode light-pen and scope devices are currently being installed, or, in the case of the film scanner, are being made ready for production.

Control Data's man-machine graphic systems are being developed at its Digigraphic Laboratory where the aim is to develop an information-processing system which would merge the best attributes of film-based graphic systems with those of digital-computer based systems. The cross-over media between the image world and the digital world evolved as the cathode-ray tube. These display images are suitable for film recording and man-image communication, and yet are easily driven by digital devices.

The term "Digigraphic"—a registered trademark of Control Data—refers to a broad range of hardware and software systems which usually involves:

1. On-line, real-time use of a computer system by a man using a cathode-ray tube with light pen and function-control buttons, frequently operating in a time-shared mode with other consoles on the same computer while working on the same, or a different, data base with the same, or different, application program.

2. Digital manipulation of a data base containing geometric descriptions, alphanumerics and logical linkage information pertaining to the geometric configuration of the image, as well as category information about the characteristics of the physical entity the image represents.

3. Extensive use of graphics and symbols as the primary man-system, conn-communication language.

The Digigraphic System 270 consists of a console utilizing a large, 22-inch flatface cathode-ray tube, a controller and a combined auxiliary memory/buffer combination. The console is equipped with a light pen consisting of a thin, 48-inch fiber optics cable which pipes light from the screen face into a photo-multiplier tube in the console cabinet. Twenty-four function buttons are used to communicate with the Functional Control Program (FCP) which controls time-shared operation and provides basic graphic manipulation capabilities. A FORTRAN-oriented interface is provided by FCP to facilitate communication with the user's application program.

FCP works with a two-level data base. The prime level, called the Digigraphic List, is created by the computer system as a direct result of light-pen and function-button manipulation. This, concurrent with manual light-pen and function-button operations, is used to generate a secondary data base which creates a graphic response on the cathode-ray tube and verifies the input commands. The geometric precision of the prime Digigraphic List is high. It is equivalent to constructing graphic images on an X-and-Y coordinate grid structure with approximately eight million intersections on a side. If the coordinate grid structure is designated as having spacing of 0.0001 of an inch between each coordinate line, the master grid is equivalent to a working area of approximately 800 square inches with an addressability down to 0.0001 of an inch. The secondary data base, used for console display and film recording purposes only, consists of a coordinate grid structure with 4,096 intersections on each X and Y axis.

The dual data-base concept facilitates standardizing the prime information-manipulation techniques by separating the output representation (CRT display, X-Y plotter, photo-recording) from the main graphic data base, the Digigraphic List.

Digital-graphic techniques are generating applications in three major areas: computer-aided design, computer-based research activities, and special types of command and control systems.

While extensive use is currently being made of computers in engineering design, most conventional computer applications are in post-design analysis and evaluation. Few existing applications involve the actual creation of a design on-line in real-time. Indeed, the extensive use of computers in the post-design role of evaluation has created many "closed-shop" computer complexes where design function users are, in many cases, forced into situations in which parameter-evaluation processing requires turnaround times ranging from many days to, at best, overnight.

The use of Digigraphics in computer-aided design is aimed at providing engineers and scientists with graphic communication and computational back-up to perform both the actual design and the design analysis and evaluation in an economical manner while on-line with a time-shared computer system.

FCP eliminates the need for a user to learn the complex technology of display programming. Users easily establish communication between the console display and their application program through simple FORTRAN statements to FCP. Without such a combined hardware and software approach, a user could become buried under a graphic-programming requirement which might exceed the complexity of the very application program it is designed to be used with.

In computer-aided design application, teams of engineers and scientists, working in separate geographical areas, work on a common geometric representation of the end item being designed. This "model" constitutes the prime data base (Digigraphic List) on which all parameter testing is accomplished by the users' application programs. The data base also contains pertinent notes, category information and geometric-linkage data so that various subsystems can be called up and displayed on the console without unnecessary clutter. Since the model is maintained in a digital form, specific views or sections can be transmitted long distances over conventional telephone lines for reconstruction and review at remote locations.

The Digigraphic List also serves as the main source for documentation on the item created:

A. It makes feasible the use of soft-copy devices (CRT's) to view the model in any desired level of detail from remote locations thus reducing the requirements for maintaining hard-copy references at end-user locations.

B. The list-structure techniques linking both graphic images of an end item with a narrative on its physical characteristics and identification number make parts-accounting an automatic by-product of the graphic creation process.

C. With only one small model, represented by the Digigraphic List, to which all end users will have immediate access, regardless of their location or the nature of the documentation they desire, engineering change-order procedures become significantly simpler.

D. By saving all changes in part descriptions or their images in chronological order, through the Digigraphic List techniques, the user may reconstruct the history of the design for reliability analysis and for field modification.

SIGNIFICANT NEW DEVELOPMENTS IN THE DIGITAL-GRAPHIC FIELD

There are two major new concepts currently maturing which will have a significant impact on Digigraphic applications:

1. Three-dimensional data-base manipulation and viewing. Expansion of the initial Digigraphic List concept from the flat-grid coordinate structure to the management of a three-dimensional cube structure will, in the future, make feasible significant expansion of the application base. The construction and manipulation of three-dimensional representations of physical entities must be considered from two aspects: (a) the management of the fundamental data base which includes the coordinate point (X, Y, Z) information about the physical entity; and (b) the creation of suitable images for viewing various projections of the physical entity within its 3D, cube-like coordinate structure.

(a). Data-base management.

The data-base model is created by engineers and scientists, working with a three-dimensional coordinate grid structure, who specify geometric parameters and pertinent notes required by the end user of the model to establish the proper linkages between related image items. In addition, the creators specify the proper categories for determining the functional or end-use classification of all items in the model.

(b). 3-D display.

It is important to recognize that in working with 3-D techniques, the secondary data-base manipulation necessary to present a 3-D perspective on the screen of a 2-D cathode-ray tube is a separate problem from the management of the 3-D data base itself. Currently there appears to exist much more know-how on how to manage and process 3-D coordinate point information and its logical linkages than there does about how to generate accurate views of this same information on a CRT console. Such demonstrations as rotating a 3-D object on a CRT may or may not provide insight into the structure of the real entity, depending upon the simplification required in order to generate the display image when, for example, the view involves complex surfaces.

2. Scanning. Precision cathode-ray tubes with exceptionally high resolution now make it feasible to, in effect, give the computer an eye. In this technique, a film clip or aperture card is placed in front of a suitable optical system and cathode-ray tube. The computer places the CRT beam in a certain X-and-Y location on the face of the cathode-ray tube. The light passes through an optical system, through the film image, and into a light detection unit behind the film. If the beam encounters a dark area on the film, this change in light transmission is perceived by the light sensor. The information for each discrete location which the computer program probes is passed back to the computer program so that it can make decisions on the next location it wishes to examine. Selective searching, based on some pre-programmed plan, soon produces enough information for the scanner program to begin creating a Digigraphic List containing pertinent coordi-

nate-point information about the image on the film. Thus, in theory, it would appear feasible to create Digigraphic Lists representing existing graphics recorded on aperture cards. In truth, however, the scanning procedure amounts to actually scaling the film image in order to obtain pertinent dimensional information. Hence, the information is not valid enough, dimension-wise, to be used in constructing a high-precision geometric model of the part described by the drawing. Since the drawing has been digitized, however, it may be displayed on a cathode-ray tube and conventional light-pen operations may be executed to produce changes in the image. This modified image can subsequently be recorded on film from the face of the same, or an equivalent, precision cathode-ray tube. As a drawing-change technique, considering the amount of capital equipment involved, this type of application is considered by many to be economically marginal. More promising applications in scanning film images exist in the analysis of photographs where precise coordinate-point information is not critical or where advantage can be taken of the ability of the system to detect tonal changes, as, for example, in the analysis of bubble-chamber particle traces, computer-based analysis of chest X-rays and in the analysis of slides containing organic tissue sections. Real-time scanning of dynamic phenomena also promises some fascinating new applications for digital computers. Here an actual occurrence is analyzed by a computer and a precision cathode-ray tube, in real-time, as it happens. The computer program examines the changing phenomenon and creates a Digigraphic List so that a human operator can subsequently view the sequence of occurrences on the face of a console CRT at a later time. Since each light probe by the scanner requires only about twenty-five millionths of a second, a considerable number of moving phenomena can be observed in real-time and later regenerated on a console CRT at a considerably slower rate for analysis purposes. In some cases, the computer program may also elect to activate a photorecording device to make a film image, at some point in time, of a continuous occurrence which it has been continually monitoring. In one such application, a computer, through a precision CRT, watches a bubble chamber and makes pictures of particle traces which are of significant interest, but ignores those traces its computations indicate are of no significance.

APPLICATION PROGRAMMING—CONSTRUCTS

In addition to these activities of Control Data, there is one other that should be of interest to this conference. It is the operation of an engineering software development subsidiary, called Meiscon Corporation, whose purpose it is to prepare useful application programs for the engineering community. Several of these program systems have been developed on an R-and-D basis to evaluate the usefulness of computers to as yet untried engineering activities. As a means of assuring that these program systems can be truly evaluated, Meiscon also performs contract consulting engineering to prove their worth. The largest program of this type to date is called Constructs.

Upon completion, Constructs will be an integrated-design and fabrication-detailing system for steel structures. The design portion of the system begins after a framing plan has been developed by the engineer. The output of the design becomes input to the detailing phase whose function it is to generate drawings equivalent to those produced manually for shop fabrication. Use of the Digigraphic capabilities has not, as yet, been integrated into the system planning, although it will be essential for the extension of the system to include the important phase of structural layout. Under current development are the following three parts of the design system.

Part 1 provides structural analysis for two- and some three-dimensional indeterminate frames with both horizontal and vertical loads. Until implementation of Digigraphics, the system requires that the engineer locate the positions, but not the size, of the structural members through card input. Both pinned and fixed connections will be included within the scope of the analysis. Resulting from the stability analysis are joint moments and shears, which are rearranged so that they will be suitable for combination with the results of Part 2, the gravity floor-load distribution program. Adequate printout procedures allow for inspections of the data to assure the user of the proper program response when necessary, and to allow the user to conveniently modify the system decisions.

Part 2 of the system distributes both live and dead gravity loads from floor to beam and from beam to column. To accomplish this, the program develops a digital model (i.e., data list) of the structure so that loads may be transmitted between all affected members of the structure. All of the loads distributed to the members from the floor will be combined with loads from the stability analysis of Part 1 to form a compilation of all loading conditions for the design of each member.

A column-schedule program will be implemented to calculate preliminary loads from bents and floors for the selection of column sections between each pair of floors. The printed results allow the engineer to set splice points and pick desirable column sections for the whole structure. It is at this point that optimizing techniques may be applied to select the least-cost column arrangement. The column schedule is produced at an early date so that the substructure design may proceed.

Part 3 of the system analyzes each member of the structure as a free body with both input and previously calculated loads applied to the member. From this loaded free-body condition, shears and moments are calculated and an economical section is selected. If the engineer chooses, he may ask for a range of members rather than for one.

Part 3 can also handle beams, columns and bracing. It is comprehensive enough to meet all of the requirements of the new AISC code. It has been implemented to provide the engineer with either final selection as to member size, or a range of feasible sizes for the engineer's selection.

AUTOMATED DETAILING SYSTEM—CONSTRUCTS

The second portion of Constructs is primarily of use to the steel fabricator. Input may be the output of the design phase or it may be independently generated. The general concept of the Constructs system is to provide by computer, program and plotter the capability of preparing structural steel fabrication drawings. These drawings require more than the simple placement of lines on paper. The engineering concepts to fit the structural material and attached secondary connection material are an important part of detailing and are included in the Constructs system. The inclusion of these detailing operations within the program allows for more efficient use of a computer and plotter by avoiding wasteful man-machine communication and thus provides maximum return on the use of the equipment.

Specifically, the Constructs system is capable of handling the main members of AISC, Type-2 steel structures. The basic input describes the general features of the structure, the specific detailing and connection design criteria, and the shape, location and support conditions of each member. After the initial input, the system is designed to operate with little, or no, user intervention or additional input. The user is only required to review error messages at intermediate points during the operation.

The Constructs detailing system can be thought of as being divided into four basic phases. Each of these phases simulates the thought processes and physical actions of an experienced draftsman. In the first phase, the draftsman studies a set of engineering drawings, assimilating and categorizing what he sees and tentatively planning what he will do. Constructs looks at a set of plans by reading punched cards and storing into memory pertinent data regarding the structure. The next phase, that of determining how the members can be interconnected, is essentially a handbook and mechanical-fitting operation for the draftsman. Constructs uses stored program logic and AISC tables, just as a draftsman draws upon his experience, to select and design connections. In phase three, the draftsman decides how to portray schematically the various pieces and dimensions of the structural members in a manner commonly understandable by the fabricating shop. Here he selects the schematic pictures, dimension lines, notes, etc., that are required to produce the complete detail. Constructs, again through stored program logic, provides these same functions. The last phase involves the physical movements that are necessary to portray the draftsman's mental picture by movements of his pencil on the paper. Constructs now generates the plotter commands which, by guiding a drafting device, produce these pictures in exactly the same manner. The details of the final drawings are comparable in all respects to those produced manually. The quality of the automatically generated drawings exceeds that of the manual, both in clarity and regularity.

The major program functions are as follows:

INPUT—The basic data for Constructs is that information portrayed on the engineer's design drawings. Data regarding each member's relative location within the framing plan (i.e., 6′ from Bent C; 2′8″ from Floor 7; etc.) are input along with a declaration of the absolute location of the structure's bend and floor plans; the user need only input what he sees on the design drawings, and does not have to make tedious and error-prone mental summations to compute coordinates. The man preparing the input need not be concerned about connection design or dimension determination since the system handles these functions.

The location of each member, its shape, size and orientation are also input. In addition, a declaration of the members which support the piece in question is also supplied. Subsidiary information regarding the shop and field methods of fastening is also input. This data is written on preprinted input forms. It is then keypunched and a magnetic tape is prepared for input to the system.

VALIDATION—An extensive series of validation tests are supplied to the input data to keep detectable errors from passing through the system. Member sizes are checked for validity against the AISC table. Abnormally long, short or skewed members are brought to the viewer's attention as possible error conditions. The support declaration made by the user is verified for consistency, and an "internal model" of the structure is created to determine the design sequence of the members. The declaration of shop- and field-fastening modes for the various classes of members are also cross-validated to avoid inconsistencies in fabrication procedures. If any error conditions are uncovered, printed messages indicating the source and cause of the error are given. When necessary, the user adds correction cards to the original input and reruns this phase until the data is correctly formed and the system acknowledges this condition.

CONNECTION DESIGN—After the input has been assimilated, it is sorted by the system into the proper detailing sequence for submission to the design phase. In this operation, each member is interrogated as to shape, location and supporting members. The individual joints are designed by the design-and-detailing procedures recommended by the AISC code and by additional parameters of detailing criteria input by the user. All the basic facts regarding each member, its end and interior connections, cuts, copes, fasteners, etc., are determined and are stored in preparation to plotting.

GRAPHIC PREPARATION—This phase interprets the stored data for each member and its associated connection material, and determines the manner in which the member should be represented graphically. Each piece of data is "looked" at and a judgment is made as to which subroutine will have to be used to provide the necessary graphic views. The cumulative effect of all the "data-to-picture" determinations provided the total detail drawing. The output of these subroutines consists of simple declarative commands such as "draw clip," "draw weld symbol," "draw hole pattern," and indicates where these actions are to take place. The effect of this phase is the same as speaking to a draftsman at some remote location over the telephone and telling him the standard drafting components which will make up the picture.

DRAFTING TRANSLATOR—Once the basic drafting components have been selected, they are then transformed into actual pen movements as required by the language of the plotter. Since each plotter has an individualized command structure, a general plotting language capable of manipulating graphic entities, such as "rotate clip," along with a post-processor for the particular plotter, has been developed. The general plotting language consists of all the basic operations any plotter would be called upon to perform.

SCOPE OF CONSTRUCTS

Constructs was developed to provide a broad ability to handle the major members of beam-and-column type structures. All of the specifications, design procedures, and rolled sections as found in the American Institute of Steel Construction, 1963 version, are incorporated in the Constructs logic. The system allows for any combination of field bolting with shop welding, riveting or bolting.

There are over 2,400 combinations of input allowed which can generate tens of thousands of picture combinations. In addition to preparing the drawings, Constructs will produce a computer-listed bill of material for each drawn sheet and a listing similar to a shop list or mill order for all members input to the system.

The drawing procedure, including both pictorial representation and dimensioning method, which is used in Constructs attempts to satisfy the majority of all structural steel detailing systems.

In addition to Constructs, Meiscon has prepared a highway design system, extended from some of the techniques of the DTM system developed at MIT. Of interest is the terrain definition technique which allows for relatively close definition of the central ground, no matter what alignment is chosen. This system produces, as output, drawn pictures of terrain cross-sections, roadway alignment and earthwork-mass diagrams. Study is currently underway to generate an even more precise model from aerial photos so that final design can be accomplished from the same terrain data.

Besides these design systems, a number of other developments are being pursued utilizing computers to generate drawings for process piping and for manipulated three-dimensional objects.

As indicated previously, the restriction of time precludes a discussion in depth of any one of the many activities being pursued at Control Data. I think this short synopsis will indicate to you that the foundation of problem-solving capability of today's engineer is already broad and is growing.

CRT DISPLAYS IN ENGINEERING DESIGN

M. A. Ford
Digital Equipment Corporation
College Park, Md.

INTRODUCTION

The digital computer is a very powerful engineering-design tool. As the cost of high-speed scientific computers comes down towards the $20,000 figure, and the speed and software performance of such computers increases to 1 or 2 μsec add-time, it becomes almost credible to think in terms of one computer for each engineer. Similarly, as time-sharing becomes more in vogue, and very large, very high-speed computers become simultaneously accessible to as many as 100 or 200 engineers, one can certainly envision the possibility of one time-shared terminal for each engineer.

The usefulness of any computer or time-shared terminal to an individual engineer is limited by the language barrier which exists between the engineer and the machine. This language barrier might be overcome by any of the following means:

A. Make each engineer as proficient in programming for problem-solving as he is in calculus.

B. Construct problem-oriented programming languages which permit basic design problems to be expressed by the engineer in a familiar language. FORTRAN is a general example of this approach.

C. Utilize a language common to both computer and engineer which is a drawing or sketch, i. e., use CRT displays with light pen and keyboard as a problem input device.

The first solution is neither possible nor desirable. In a scientific world where there is an information explosion taking place, it seems to defeat the purpose if all engineers have to become programming experts in order to realize the advantages of computers. The second solution to the language barrier is a practical one and one which is, in fact, being used today. However, its practice does not take full advantage of the computer as a design tool. Use of problem-oriented languages implies the use of the computer as an expensive slide rule. A particular problem is given by the engineer to the computer and the computer calculates the solution or solutions. The engineer must then place this solution in the context of his design and decide what to do next. In practice, this may result in a new set of parameters and another pass through the computer. The feed-back time most likely is on the order of days. Whereas this is a vast improvement over actually using a normal slide rule, it is hardly instantaneous feed-back.

The third solution to the language barrier has the very interesting side benefit of providing a means of instantaneous feed-back. A designer may sketch his structural steel assembly on the face of the CRT and place test loads where he wishes. If the stresses on the individual beams are not to his liking (a fact he can ascertain right then and there by reading the values displayed next to the beams), he can change the design with the light pen and keyboard until he sees that his stress requirements have been met. This on-line design feed-back or design evaluation is possible with a CRT display terminal connected to a computer.

In fact, only by means of a CRT and light pen (or an equivalent interactive I/O device) can the man-machine language barrier be overcome and the computer truly be used by the engineer as a design tool. Without such I/O equipment, the computer is no more than an expensive slide rule.

Where does the Digital Equipment Corporation fit into this philosophy? Strangely enough, DEC has been manufacturing and supplying CRT systems to the engineering and scientific community for over five years. In fact, DEC was the first computer manufacturer to sell CRT displays to the non-military market and is now one of the leading suppliers of display systems to scientists and engineers. In addition to CRT systems, DEC also manufactures a line of small scientific computers.

THE PDP-8

As I mentioned earlier, there are two possible means by which an engineer can have hands-on access to a computer: a time-sharing terminal, or a small scientific computer. DEC manufactures such a computer which is called the Programmed Data Processor-8 (PDP-8).

The PDP-8 makes available to engineering, scientific, and educational users a compact but complete general-purpose digital computer with a high-speed, random-access, magnetic-core memory. It is intended for use in data processing or as a control element in on-line data handling, experiment monitoring, or a process-control system. The standard PDP-8 is a complete hardware and software system, ready to perform general-purpose computations and/or control operations on the day it is delivered. Optional data-processing peripheral equipment or special control devices are easily connected to the PDP-8 to form a special-purpose system. Instruction format is simple and logical.

One instruction format exists for each of four groups, so that format of individual instructions need not be memorized by the programmer. A complete software system accompanies each hardware system and contains FORTRAN, MACRO-8 assembler, a library of mathematical subroutines, and utility and maintenance programs.

The PDP-8 performs binary operations on 12- or 24-bit 2's complement numbers. The 1.5-µsec cycle time of the machine provides a computation rate of 333,333 additions per second. Addition is performed in 3.0 µsecs (with one number in the accumulator) and subtraction is performed in 6.0 µsecs (with the subtrahend in the accumulator). Multiplication is performed in approximately 315 µsecs by a subroutine that operates on two 12-bit numbers to produce a 24-bit product, leaving the 12 most significant bits in the accumulator. Division of two 12-bit numbers is performed in approximately 444 µsecs by a subroutine which produces a 12-bit quotient in the accumulator and a 12-bit remainder in core memory. Similar multiplication and division operations are performed by means of the optional extended-arithmetic element in approximately 21 and 37 µsecs, respectively.

Flexible, high-capacity, input-output capabilities of the computer allow it to operate a variety of peripheral equipment. In addition to standard Teletype and perforated-tape equipment, the system is capable of operating in conjunction wth a number of optional devices such as high-speed perforated-tape readers and punches, card equipment, a line printer, analog-to-digital converters, cathode-ray tube displays, magnetic drum systems, and magnetic-tape equipment. Equipment of special design is easily adapted for connection into the PDP-8 system. The computer is not modified by the addition of peripheral devices.

PDP-8 is completely self-contained, requiring no special power sources or environmental conditions. A single source of 115-volt, 60-cycle, single-phase power is required to operate the machine. Internal power supplies produce all of the operating voltages required. FLIP CHIP modules utilizing hybrid silicon circuits and built-in provisions for marginal checking insure reliable operation in ambient temperatures between 32 and 130 degrees Fahrenheit.

The low price ($18,000 and up) and high performance capability of the PDP-8 certainly places it in a position of consideration for any engineering design requirement where large-computer power is not required. In fact, it becomes economical to have on hand such a computer with FORTRAN capability for the purpose of experimentation with design-problem solution techniques.

THE PDP-7

The PDP-8 is a 12-bit machine and thus has some limitations as to programming ease and computation precision. Although double- and triple-precision arithmetic routines are available when needed, the 18-bit PDP-7 is more suitable if a large portion of the engineering computation requires such precision.

DEC's Programmed Data Processor-7 (PDP-7) is a general-purpose, solid-state, digital computer designed for high-speed data handling in the scientific laboratory, the computing center, of the real-time process control system. PDP-7 is a single address, fixed 18-bit word length, binary computer using 1's complement arithmetic and 2's complement notation to facilitate multiprecision operations. Cycle time of the 4,096-word, random-access, magnetic-core memory is 1.75 µsecs, providing a computation rate of 285,000 additions per second.

The basic PDP-7 includes the processor (with operator console); 4,096-word core memory; input/output control with device selector (up to 64 I/O connections); information collector (seven 18-bit channels); information distributor (six 18-bit channels); program interrupt; data interrupt; I/O skip facility; I/O status check; and real-time clock. A high-speed paper-tape reader (300 cps), high-speed paper-tape punch (63.3 cps), and KSR-33 teleprinter (10 cps) are standard input/output equipment with the basic PDP-7.

Interface to the PDP-7 allows fast parallel information transfer between the computer and a variety of peripheral equipment. In addition to the teleprinter, keyboard, and high-speed perforated-tape reader and punch supplied with the basic computer, the PDP-7 optional peripheral equipment includes magnetic-tape equipment, card equipment and line printers, serial magnetic-drum storage, cathode-ray tube displays, a data communication system, and analog-to-digital converters. Special purpose I/O equipment is easily connected using an interface of standard DEC modules.

CRT DISPLAYS - THE 338

The 338 Programmed Buffered Display is a precision, incremental display system. There are three elements in the 338: The PDP-8, the display processor, and the display tube and its associated electronics. The combined capabilities of this buffered display with processor offer the design engineer an unusual degree of versatility and economic advantage over conventional buffered displays.

Interactive console

As I mentioned before, the language barrier is really overcome when the designer can receive instantaneous feedback from a change in design and can readily express this change in design by using the light pen and keyboards connected to a display processor to alter his design picture. The 338 may be used extremely effectively as an interactive display input/output device, either connected to a larger computer, or standing by itself. The existence of the PDP-8 processor between the display and a larger computer offers several other advantages:

1. Hardware interface costs are reduced since the job of controlling the interface between the 338 and the larger machine can be accomplished by PDP-8 programming, rather than by more expensive hardware control logic.

2. Routine tasks such as light-pen following, character generation, rotations, and elementary-data structuring can be handled by the PDP-8 programming, eliminating frequent interrupts to the main computer.

3. In addition to these advantages, the 338 offers even more usefulness in the case of remote display terminals to a large time-shared system. Since most telephone line connections between remote terminals and large computing centers have a very low bandwidth, it would take almost a minute to transfer a complicated display file from the central machine to the display if the information were formatted for the display prior to transmission to the remote display. If the PDP-8 processor can be used at the remote terminal to appropriately format the display data as it comes off the line, the quantity of data and hence the transmission time can be reduced by several orders of magnitude.

These are some of the reasons why DEC feels that the 338 is extremely useful to the user who is concerned with computer-aided design. Let me now describe the 338 in a bit more detail.

338 Details

The 338 Buffered Display System permits rapid conversion of digital computer data into graphic and tabular form. Its combined capabilities offer the user an unusual degree of versatility and accuracy.

A self-contained unit with built-in control and power supplies, the 338 requires only logic-level inputs for operation and may be easily connected to any digital system as a buffered display with processor, or it may stand alone as a powerful computer-driven display system. Location of any desired point may be specified by any of the 1024-X and 1024-Y coordinate addresses contained in a 9-⅜-inch square on the tube face. Discrete random points will be plotted at a rate of 35 μsec per point, in point- or graph-plot mode. If the sequences of points lie near one another, the plotting rate will be about 6 μsecs per point. In the other modes, the plotting rate is 1.2 μsec per intensified point. Nonintensified points, however, are plotted at a rate of 250 μsec per point. Magnetic-deflection and focusing techniques result in uniform dynamic resolution over the entire usable area of the tube face and maximum spot size of approximately 0.15 of an inch. Construction is solid state throughout with excellent stability.

The 338 is a stored program system that was designed with the programmer in mind. Some of the display capabilities are:

1. **Data-acquisition cycle stealing** - The display receives 12-bit data and control words from the PDP-8 memory via the PDP-8 data break channel. The data break channel is a high-speed (1.5 μsec/word), direct-access channel that passes words to the display transparently to the program in execution.

2. **Automatic scissoring** - The display can be programmed to represent a 9-⅜-inch square window viewing a 75-inch square drawing. This window can easily be moved around the drawing by simple translation of coordinates. Only the area of the drawing corresponding to the display window will be seen.

3. **Multilevel subrouting** - The control state permits the display to jump from accessing one location in the PDP-8 memory to any other. When it is desired to jump to a display subroutine, the return address is automatically stored in a push-down list.

4. **Man-machine interaction** - The use of push buttons allows the operator real-time control of the display and computer programs. The high-speed light pen permits the operator to establish a reference within his program to a specific point displayed on the screen. The full duplex TTY Model ASR-33 provides a keyboard and printer, and also a 10-cps paper-tape punch and reader.

5. **Character generation** - In addition to line generation, the display hardware can display characters specified by 6-bit codes. Each character is displayed in an average of 22 μsecs.

6. **Communication with display registers** - The contents of the X-Y position-registers and the display address-counter can be read into the accumulator of the PDP-8 via IOT instructions. Likewise, the PDP-8 can load certain registers in the display which are used for starting the display or setting up initial control conditions.

Display modes

The display logic is divided into the control state and the data state. In the former, the 12-bit words are decoded as instructions to the display logic. These instructions can set the parameters and mode which affect how the data is displayed; or they can cause a jump, jump to subroutine, or skip on display flags. Data-state words direct the deflection amplifiers as to the number and direction of mode. Examples of data modes are: increment mode, point mode, vector mode, and graph-plot mode. Each data mode is also equipped with an escape mechanism to return the display to the control state.

In summary, the 338 Buffered Display System is a powerful scientific computing system by itself, or an economic terminal for a larger computer system. In both applications it affords the engineer an opportunity to present his problem to the computer in a most efficient way, and to have his results indicated to him in a most meaningful way. This truly represents elimination of the man-machine language barrier and opens the way to effective use of the digital computer as a design tool.

PROSPECTS AND REQUIREMENTS

SECTION F

CHAIRMAN: N. M. NEWMARK, *Head, Dept. of Civil Engineering*, University of Illinois

COMPUTERS IN ENGINEERING DESIGN vs. DESIGN OF COMPUTERS

L. A. ZADEH
University of California
Berkeley, Calif.

I regret very much that the necessity of attending another meeting has prevented me from coming here earlier and listening to the many informative papers which have been presented on the subject of the impact of computers on engineering design. I believe I can take it for granted, however, that the Conference has established beyond any doubt that the advent of the computer age is having a great impact on engineering design, and, consequently, on engineering education.

The acceptance of this fact does not imply, however, that we have a clear view of the role of computers in engineering education, nor does it settle the question of what place, if any, courses in engineering design—whether computer-oriented or not—have in engineering curricula. I may be making myself a persona non grata at this Conference by raising the latter question. Be that as it may, I must confess to some doubts that design can be taught effectively in the classroom, even when it is heavily spiced with computers. My doubts are based on the fact that realistic engineering design requires a great deal of know-how, and that, in an era of rapid technological advances, know-how can be acquired only through practice and experience, and not through teaching in the classroom.

As I see it, there are roughly three schools of thought in regard to the place of design courses in engineering curricula. First, there are those educators who argue that, because of the rapid depreciation of specialized engineering training brought about by changes in technology, engineering curricula should aim primarily at providing the student with basic tools of long-lasting value, such as mathematics, physics, chemistry, mechanics, thermodynamics, system theory, etc., and only secondarily aim at giving him competence in specialized subject areas in his major field of study. Implicit in this view is the assumption that the training of an engineer does not stop upon graduation, and that the know-how and experience needed for design can be acquired much more efficiently in an industrial rather than an academic environment. Second, there are those who argue that we should go beyond providing our students with a broad training in the fundamentals and give them some exposure to design in various specialized subject areas, e. g., in courses on electronic circuits and devices, thermal systems, nuclear reactors, etc. Finally, there are those who feel that we should go much farther than giving a little flavor of design in various engineering courses and should make a concerted effort at teaching design per se, realistically and systematically. I take it for granted that this point of view is well represented at this Conference. As for myself, I fall somewhere between the first and second of the above groups, since I am far from convinced (perhaps I would be more convinced had I listened to the papers presented here during the past two days) that design is a teachable subject.

Having defined my prejudices, I should like to make a few remarks concerning the role of computers in engineering design and then turn to a question that is closer to my heart, namely, the training of engineering students in computers and their use.

Extrapolating present trends, it appears certain that in a few years' time engineering design will become a highly computerized process with the design of a standard system reduced to locating an appropriate program in a file or a library, feeding the system specifications to the computer and letting it produce a detailed design or, perhaps, instructions to computer-controlled tools or assembly lines. In the case of a design of a non-standard system, the computers are likely to be used in an interactive mode, with the designer employing the computer in a sequential manner (probably on a time-shared basis) involving back and forth flow of information and commands between the computer and the designer. This, of course, is an oversimplifid view of the future, but it will suffice for purposes of our discussion.

Unquestionably the computer-based methods of design will be much more powerful and effective than the techniques employed today. Does this mean, however, that design will become a teachable subject? Not necessarily, in my opinion, since it is—and will continue to be—very difficult to abstract from the varieties of specialized design procedures a body of concepts and techniques with wide applicability. Without such a body of concepts and techniques, engineering design—whether computer-oriented or not—could not qualify as a subject that could or should be taught in the university.

What effect, then, will the computerization of design processes have on engineering education? Clearly, all engineering students, regardless of their field of specialization, will have to be proficient in the use of computers as information-processing systems. What is not clear, however, is the extent to which engineering students should be taught not just how to use computers as aids to design, but also about their internal structure, about the principles of numerical and non-numerical programming, about automata theory and formal languages, about heuristic programming and artificial intelligence, and, more generally, about an array of subjects which comprise what are currently referred to as computer sciences.

This question is not yet a pressing one in fields other than electrical engineering—a field which bears a special relation to computer sciences by virtue of its long history of involvement with communication and information processing. Indeed, all large-scale information-processing systems are, in the main, electronic in nature. This is a consequence of the fact that, in an information-processing system, a datum is represented as a state of one or more of its elements. Thus, to process information at a high speed it is necessary to have a system whose states can change rapidly, which in turn requires that the information-storing elements of the system have low inertia. These considerations explain why electronic components and circuits having extremely short transition times from one state to another play such a basic role in high-speed information processing.

By virtue of their special competence in the design of electronic systems, electrical engineers are naturally concerned not only with the use of computers—as are all other branches of engineering—but, more importantly, with their design and operation. However, although no one questions the central role of electrical engineering in computer hardware, there is a lack of unanimity on the extent to which it should concern itself with computer software. There are some—and I fall into this category—who argue that, because of their long-standing involvement in information processing, electrical engineering departments should greatly expand their teaching and research activities in computer sciences and give their students a thorough preparation for careers in this and related fields. There are others who maintain that computer software should be the responsibility of computer science departments, of which several are now in existence and more are being set up. It may not be inappropriate at this point to mention that a committee called the COSINE Committee has recently been set up under the sponsorship of the Commission on Engineering Education (I see in the audience, Newman Hall, its Executive Director, who was instrumental in organizing the Committee and providing it with support by the Commission) to aid and guide electrical engineering departments in the strengthening of their program and course offerings in computer and information sciences.

It is interesting to speculate on the effect which the establishment of a computer science department at a university is likely to have on the electrical department at that university. In my opinion, the two can coexist if there is a framework for cooperation between them in the form of joint appointments and a reasonable division of responsibilities for teaching and research in computer sciences and related fields. However, at those universities in which the main responsibility for training in information processing will be in the hands of computer science departments, electrical engineering—divested of its activities in information processing, except possibly on the hardware level—is likely to be a much less important and viable field of study than it is at present and has been in the past.

Putting aside jurisdictional questions, I should like to draw attention to an area in computer sciences, namely, heuristic programming and artificial intelligence, which is certain to have an appreciable impact on engineering design in the years to come. Some of you may be puzzled somewhat by my mentioning in one breath artificial intelligence and engineering design, since, on the surface at least, there does not appear to be any connection between them. In fact, I believe that there is a very close relationship between the two and that heuristic programming and artificial intelligence will—in the not distant future—play an important role in engineering design.

Let me explain why I feel this way. In recent years, engineering and, in particular, electrical engineering, has tended to become increasingly abstract and mathematically oriented. This trend is especially pronounced in such areas of electrical engineering as system theory, information theory, control theory and network theory, with the result that most workers in these areas, whether in universities or industrial research laboratories, tend to concern themselves only with those aspects of system design and analysis which are susceptible of mathematization, that is, of yielding provable assertions in the form of lemmas and theorems.

Unfortunately, the real world we live in is much too complex and fuzzy to admit of being fitted to a precise mathematical framework. In practice, this means that there are many basic problems in system design and analysis which are beyond the reach of classical mathematical techniques, and that we should strive to develop other approaches—not necessarily appealing to a classical mathematician—

for their solutions. This thought was expressed very succinctly by Professor J. Tukey of Princeton University and Bell Telephone Laboratories, when he said, in paraphrased form, "What we need now are not so much precise solutions to approximate problems as approximate solutions to precise problems." Clearly, in saying this he implies that too many of us have been and are busy solving artificial problems which bear little relation to reality and that it is time for us to get down to the less intellectually satisfying but more challenging problem of devising approximate approaches to the solution of realistic problems.

It is at this point that heuristic programming and artificial intelligence make contact with engineering design, since, in essence, they represent an attempt at using computers to "solve" problems which do not admit of precise mathematical treatment or do not have computationally feasible and mathematically rigorous solutions. For example, many of the problems in pattern recognition, theorem proving, game playing (checkers and chess), information retrieval, system identification, machine translation of languages, etc., fall into this category. The common characteristic of these problems is that they involve in an essential way the process of abstraction, that is, the process of deducing a property which defines the members of a class of objects. It is our lack of understanding of the subconscious ability of the human mind to perform abstractions which are not well-defined mathematically that is at the root of our inability to make substantial progress toward the solution of the problems cited above.

A heuristic program is, as its name implies, a less than rigorous or invariably effective algorithm for solving a particular problem. Generally, it is a program that is arrived at by heuristic reasoning and is not guaranteed to always yield a solution, when one exists, or give a solution that is optimal in some specified sense.

I believe that the really challenging problems in engineering design in the future are likely to involve the design of large-scale systems such as air-traffic control, information retrieval for libraries, banking, space communications, etc. Heuristic programming is certain to play an important role in the design of such systems, because they are much too complex for analysis, not to speak of synthesis, by precise algorithmic techniques. It is for this reason that I believe that, although heuristic programming and artificial intelligence do not have as yet much to contribute to engineering design, they will, eventually, be accorded an important place in the design of large-scale systems of various types.

In conclusion, unlike most of you, I have some doubts that courses in engineering design, whether computer-oriented or not, should be included in engineering curricula; I am convinced, like most of you, that all engineering students should receive substantive training in the use of computers; and that electrical engineering students, in particular, should receive specialized training in computer sciences and related fields, so that they could not only make effective use of computers in their work, but also serve as designers of computers and computer-like information-processing and control systems.

ROSENSTEIN: *While Zadeh questions whether there is anything to teach in engineering design, his comments have crystallized some misgivings that I have had for some time. I now wish to question whether there is any such thing as computer science. By your definition, what we consider computer science is 80 per cent engineering, (and I hope you will concede for the purpose of discussion that engineering is not science) and the other 20 per cent is mathematics. I don't pretend to be a very sophisticated mathematician but I do remember reading Bertrand Russell who said, "There are no truths in mathematics."*

ZADEH: *In the first place, let me say that computer science is a somewhat unfortunate term. When you use it, it gives you the impression that this set of subject areas has to do primarily with the computer. I think that this is not really the case and a better term might be information sciences, or information processing. Computers as we know them today are just one example of information-processing machines, and, in the future, transmission and processing of information will be playing a tremendously important role in our technology and in everyday life, including perhaps the kitchen, too. Let's put aside the term computer sciences for the moment and say that there will be a complex of subjects tending to do with processing, generation, and transmission of information, and, if you take that field, I think you will find that a relatively small part of it is susceptible of mathematization. Now there are a few subject areas of mathematical nature, like automata theory, formal languages, and so forth. But there is something about these fields in which there is discreteness as contrasted with continuity which makes them less susceptible to mathematization and it is for this reason that it seems to me that engineers are very well suited to work in those areas. If you look at a typical course, let's say a course in programming systems, it is really very much an engineering course. In fact, it is so much of an engineering course that it sticks out like a sore thumb because in it there* *isn't a single equation, or theorem, or proof, nothing of that kind. It is simply a matter of describing ingenious ways of putting together any information-processing system.*

I think that it is clear that that set of subject areas will be playing a very important role, but there is some question as to the extent to which engineering students should be exposed to that field; should they be given just a smattering of training, or should they be given quite a bit? And I don't really have an answer to this question in my mind.

ROSENSTEIN: *I do agree with you about information. I believe it is one of the more important subactivities of the design process and we should teach it as such.*

SAUNDERS: *My question has to do with large versus small systems. I'd like to ask Zadeh if he sees any really basic difference in the exercise one goes through in designing a very small system such as, say, an oscillator on the head of a pin, versus the design of a very large system such as a global communication system. Is there really any basic difference in your mind between the intellectual exercise that the engineer performs?*

ZADEH: *Yes, I think that there are basic differences. As a system becomes larger and larger, the chances are that it will include a larger and larger number of different types of physical systems. It may include mechanical or electrical systems, or there might be economic factors, and so forth. In other words, as you increase the size of a system, you also increase the variety of its components. So that makes it more and more essential to be able to deal with large-scale systems without going into the physics of the components, and to deal with the components essentially on a black-box level. There is an inevitable tendency on the part of people who are concerned with large-scale system design to be abstract, without questioning what is inside the black box or what is the physical nature of the variables involved. I think somebody who designs a coffee pot is likely to be much more conscious of the fact that he is dealing with heat, electricity, materials, stresses, strain, etc., than somebody who designs an air-traffic control system, or an automatic ticket-reservation system.*

Another point is that in the case of large-scale systems, there is a crucial problem of how to pass from the overall objective of the system to objectives for the components, so that if you have a large system which has a certain function to perform (let's assume that function is well-defined), then the question arises, "How does this well-defined objective influence local objectives for the components of this system; how do you set specifications for the components in such a way as to realize specifications for the system as a whole?" This is a problem that is present in the design of subsystems also, but there it is much less acute.

I could go on listing more differences, but I think I'll stop at these two.

KARNAUGH: *I'd like to second Zadeh's plea for more understanding of large systems. One of the features about large systems which perhaps has not been emphasized very much is the practicality of the computer heuristics or algorithms that are applied. My recent experiences in this field indicate that most of us need a great deal more experience with numerical analysis programs. The ones we have very often don't work when they're needed. Not only are there bugs in the programs, but the person using them does not really understand their limitations, and when you try using half a dozen or more of them in tandem, in some kind of iteration, you have nothing but wasted time and trouble. One of the things I feel the engineering community as a whole really ought to work on is satisfactory libraries of such programs with rather clear statements about the conditions under which they may work, and tests to see whether they meet these conditions.*

PROLEGOMENA TO A SYSTEMS ANALYSIS OF EDUCATION

R. E. Machol
University of Illinois at Chicago Circle
Chicago, Ill.

I understand that the subject of this conference is education, and I should like to talk about that for a little bit. As you will see, computers will become involved at the appropriate time.

At the beginning of the present century, men were attempting to design flying machines, and some people copied the techniques that birds had used, with flapping wings which were appropriately warped to change their aerodynamic characteristics. Some of you may have seen movies of these ornithopters, which seem quaintly amusing. It was shortly recognized that technology had made available devices different from those which had been made available to birds. I assert that education is in a state analogous to the ornithopter. We have been excessively traditional in accepting the man-to-man system of education which has been handed down from past ages.

Two or three thousand years ago, the only feasible method of education was a group of students listening to a master who addressed them on the subject of his personal knowledge, and who might occasionally make a drawing in the sand to illustrate a point. Subsequently a vertical surface, such as the blackboard, became available to replace the sand, but we retain the same basic environmental structure of a classroom in which there are a body of students being lectured to by a teacher. Our modern educational system still consists basically of this classroom environment, supplemented by something called homework which is comparatively distinct from classroom study. To the best of my knowledge the only new technique which has had a significant impact on education is the so-called objective test, which may in some cases be machine scored; and there is serious question in my mind whether this is not, on certain occasions at least, a step backward rather than forward.

The educational system is an enormously vast and complex one. The largest industrial complex in our nation, the American Telephone and Telegraph Company (which incidently invented Systems Engineering at the Bell Telephone Laboratories) has a total capitalization which is less than the *annual* expenditure on education. The only system comparable in size is the total defense system of the United States, which of course uses thousands of systems engineers; and while it is comparable in dollar cost, it is very much smaller in terms of the number of people involved.

I should like to examine the educational system de novo, as a systems engineer. Normally we should first examine the functions to be performed, but these are a little complicated and we will touch on them below. Education is a communications problem, and as such we should look at the significant communications parameters.

The first of these parameters is bandwidth. Of course there is a significant difference, as every experienced teacher knows, between the rate at which we teach and the rate at which our students learn. Our speech contains information at the rate of a few bits per second, which must be reduced by redundancy due to our repetition of significant points. Actually the performance of a good teacher includes a good deal more than this, in terms of his inflections, emphasis, movements, and the like, all of which add, at least in some cases, to the value of the educational process. On the other hand, the student probably does not learn in excess of one bit per second—anyone who thinks that the average student learns more than three thousand bits of new, significant information in the classroom lecture of one micro-century duration is unduly optimistic. In any case, the bandwidth requirements are modest; with television, magnetic tapes, films, or even a comparatively slow computer output mechanism, there is more than enough bandwidth available.

Next we should look at S/N, the signal-to-noise ratio. Noise in this communications process takes many forms: it may be an instructor mumbling into the blackboard, his back to the class; a carpenter working next door; or a colleague whispering to the student about another matter. Mostly, however, it is simple inattention. How many of us have ever looked at a class when we were discussing a technical point and seen one hundred per cent alertness, with bright eyes and eager faces—or even as high a level of attention as we receive when we are telling them what questions are going to appear on a test? The fact is that it is difficult to concentrate for fifty minutes on new material, much less to do this for many hours a day, day after day. This is one of the reasons for the high level of redundancy in teaching, although perhaps a more important one is that the understanding of concepts (as distinguished from the memorizing of facts) generally requires that the point be made over and over again. This need for repetition leads to one of the fundamental inefficiencies of the present educational process, because some students require the point to be repeated less often and others more, with the result that we bore some and confuse others. This is not merely a question of the bright students versus the dull ones, although of course this is a significant factor. It is also because the students have had different exposures and backgrounds; in particular, for example, some of them have had other courses which are peripheral in nature, but none the less pertinent to a particular point.

What is the transfer function of a student? The input consists of facts, together with relationships among them and appropriate explanations and emphases which hopefully lead to "understanding" and, in the words of William Linville, "portable concepts." The output is somewhat more difficult to define or to measure. It consists of some process which takes place between the ears of the student, and which we identify by such words as "retention," "comprehension," and "understanding." There is a famous professor who had a committee meeting, and therefore left on the podium in his classroom a tape recorder with a little note to the effect that it should be turned on at the beginning of the hour and that a fifty-minute lecture was present on it. Coming eagerly to the same classroom two days later to find out how his experiment had worked, he found on the seats in front of him a whole series of tape recorders on each of which was a little note saying that it should be turned on at the hour and that there was ample tape to record a fifty-minute lecture. There is also the famous definition of a lecture as the process by which the notes of the professor become the notes of the student without passing through the minds of either. And finally I should like to tell Professor Newmark's story, about distinguishing the graduates from the undergraduates in a classroom which contains both. On the first day you walk in and say "good morning"; the undergraduate students reply "good morning," but the graduate students, who have had a little more experience in the school system, immediately write it down in their notebooks. At any rate, while we may have made little attempt to define the output of the teaching process, we have made many attempts to measure it. These measurements are called "examinations," and unfortunately their primary purpose in most cases has been to measure retention rather than comprehension. I shall return to this point below, but I wish to stress here that what we really need are what the computer man calls "diagnostics." The analogy is really very close. We need to locate error patterns and make appropriate repairs.

The next question concerns feedback, a significant factor in the control of any complex system. Feedback at the present time is given largely through tests and through correcting of homework (when homework is in fact corrected). Even when this is done, the feedback is much delayed, and we know from the work of B. F. Skinner the extreme importance in learning of rapid feedback or, as he terms it, "reinforcement."

Finally we should talk about turnaround time. This is an important measure of effectiveness in any computer operation, and in this connection I should like to make a comment concerning remote-access, time-shared computers which everybody is now talking about. It is well known that there is a sharply convex relationship, perhaps quadratic, between the performance of the computer and its cost; that is, if one pays twice as much for a computer, one gets perhaps four times as much performance out of it, or, putting it differently, one gets twice as much performance from one large computer as from two small ones of equal total cost. I recently heard the assertion that remote-access time-sharing is not going to work out as well as everyone thinks, because the advantages of the large remote computer will be lost through the disadvantages of the numerous input-output channels. This logic is based on the fact that at the present time the central processor is a comparatively small part of a total computer installation. I think this misses the point. The major benefit of remote-access time-sharing is not economic so much as it is the very quick turnaround time. For the large bulk of computer usage, which consists of small problems, turnaround time is far more significant than other factors, such as whether the computation takes two seconds or 500 milliseconds. What we are trying to minimize is a function which depends primarily on the overall time from the occurrence of a problem to receipt of the answer, with some modifications involving convenience and effort. If one has to walk to a distant computer center, come back because he finds he forgot to get a charge number, go back to the computer center and leave his cards, come back the next day to find they wouldn't run because one of the control cards was incorrectly punched, and so forth, one is not going to use the computer very much.

As far as educational uses of the computer are concerned, we must similarly optimize the overall man-machine interaction. Most student use of the computer falls into the class of small, one-shot programs written in FORTRAN or a similar language, and there is a tremendous opportunity for much greater utilization of the computer if only rapid turnaround time could be achieved. Turnaround time as it applies to the educational system that we are examining has significance when it concerns the length of time between a question popping into a student's mind, or a mistake being made, and the opportunity to obtain an answer or a correction. It is true that in so-called recitation sections students are supposed to have the opportunity to ask questions, but in many recitation sections such questions are not welcomed; even where they are, the majority of students do not ask them. In large lectures, of course, there is no possibility of questioning, and in the homework (which theoretically represents more than half of the student's learning time) the turnaround time is at least a day or two at a minimum.

Based on the above criticisms of the present educational system, I wish to recommend what I consider a better solution or system design. This would be based on an integrated set of lectures, books, and computer programs. The lectures would be of the "canned" variety, on magnetic tape or possibly movie film. The lectures can be corrected or changed, as experience dictates, or updated, by splicing in appropriate bits of tape. Of course there is no necessity for a lot of students to sit simultaneously in a single classroom with many television receivers staring at them, as is frequently done today for reasons of tradition. We visualize a student in a carrel watching a lecture, with the capability of stopping it and repeating any part which he wishes to go over. The books might be conventional text or programmed text, but in either case would be thoroughly integrated with the lectures and computer programs and designed to be used with them. The computer programs would be in the form of dialogues, between an individual student and the computer, of a type that has been frequently worked out in the past. See, for example, the discussion on PLATO (IRE Transactions on Education, Vol. E 5, pp. 156-167, 1962). See also a fascinating dialogue between a computer and a physician studying diagnosis in *Computer-Aided Instruction*, John A. Swets and Wallace Feurzeig, *Science*, 150, 527-576 (1965).

This type of dialogue is not new. Turing wrote an incredible example of one such dialogue many years ago ("Can a Machine Think," A. M. Turing, *Mind*, 1950).

It should be clear that this system would satisfy all of the requirements we have listed above. There is plenty of bandwidth; the signal-to-noise ratio is unlimited; we always have the appropriate error-correcting codes in the ability of the student to go back with his lecture, his book, or his dialogue with the computer; there is immediate reinforcement; the amount of redundancy is optimum; for each individual there is ample feedback and control of the process; and turnaround time is minimal.

Now let us talk about some of the possible objections to this system. The first objection that will be raised, of course, is that of motivation. This argument presupposes that the student is primarily motivated by a personal contact with a teacher in a classroom. But let us remember that in our great schools the bulk of the contact which undergraduates have in classrooms is with graduate students who are only a few years older than they are. It is true that we have professorial people doing most of the teaching in the smaller colleges, but we tend to look down on these smaller colleges and feel that the student is not getting as good an education there. If a really great professor gives a really great lecture, it seems a shame that only twenty students (or at most a few hundred) should hear it. Why can this lecture not be recorded and presented to thousands or millions of students? There is undoubtedly a certain magnetism in being in the same room with a flesh-and-blood speaker, but I consider that the advantage thus represented is far outweighed by the advantages of a great scholar and teacher giving a thoroughly prepared lecture with first-rate teaching aids, as against what happens today in the great bulk of our classrooms. Furthermore, we will tend to avoid the boredom of the student who is ready to proceed faster and the confusion of the student who has to proceed more slowly, and boredom and confusion also have significant effects on motivation.

What about questions? I repeat, "What happens now?" The student cannot ask questions in large lectures, he cannot ask questions when he is doing his homework (and there are two hours of homework for every hour in class), and even in the so-called recitation sections very few students do ask questions. I think under this new system we can have greater accessibility to questions by the students. Teachers who are freed from the tedium of giving the same course over and over will be available as consultants for the students who have reached an impasse. It is somewhat of a principle of system engineering (see H. H. Goode and R. E. Machol, *System Engineering*, p. 314) that automatic systems should have a provision for manual intervention in order to obviate the necessity for complicated and/or expensive provisions for unforeseen events or those of low probability. When the student gets into trouble in his computer dialogue, he presses the "Help!" button, and gets aid from the teacher-consultant. And, incidentally, this kind of consultantship, which would be the principal teaching function for most of the faculty, may keep them in closer contact with the students than the present method; that is, it will improve the feedback to the teacher.

What about testing? There are three basic functions of the examination: motivation, assessment, and feedback. Motivation is not an insignificant function; we know that many students will study harder when they know they are going to be tested. Assessment implies that we have to decide which students will get the financial aids, the honors, the admissions to our better schools, the better jobs after they graduate. Feedback is, however, the significant function of testing. This kind of testing can be very much better done in the programmed text or the computer dialogue. Presumably the student would be allowed to "cheat" on all such testing, but he would be cheating no one but himself. Occasional testing for motivational and assessment purposes could be performed in conventional ways; possibly these could be standardized on a large scale, in the manner of the present College Entrance Examination Boards, or the Graduate Record Exams.

And advisement? I do not think advisement is done very well now (for example, we are continually finding students who have failed to take their physics sequence and so are unable to begin their engineering courses), and I do not think it will be very good under the proposed system. However, I do not think it would be any different under the proposed system, and if anything it should be better. Good advisement is based primarily on the contact between an interested student and an interested instructor; because so much of the work will be done on an individual basis, it will probably be necessary to schedule more frequent interviews between the student and his advisor.

Discipline? We should face the fact that discipline is now one of the significant factors in the classroom situation. The teacher must be there to ensure that the students are paying attention and are keeping out of trouble. I recognize that this will continue to be a significant function. I believe we should hire professional disciplinarians rather than fragile Phi Beta Kappas to perform this work. They would be more efficient, and certainly far less expensive.

Finally, since this is a conference on engineering education, we must refer to the question of laboratories, which are a significant portion of that education, and so we must ask, "What is a laboratory?" It is a modeling, to reinforce and relate learning to the real world. There is a significant danger that the student in a lecture course in a scientific discipline will not recognize that what he is learning must be related in his thought processes to what is real. A friend of mine once taught an introductory astronomy course, which many students were taking because it was the easiest way to satisfy a "science requirement." One of the students in this category was a 55-year-old woman. On a certain test, my friend gave them appropriate data and asked them to compute the age of the universe. This woman came up with the answer of 37 years. My friend asked her if she was not in fact older than 37 years herself, and she admitted with a blush that she was. He then asked her if this did not appear to her to be inconsistent with her answer for the age of the universe, but she could see no such inconsistency. I realize that it is desirable to have laboratories to correct this type of situation. I realize that we occasionally get students into an electrical engineering laboratory who have never before realized that it takes two wires to carry most types of electrical signals or power. Nonetheless, I assert that a laboratory is a simulation of the real world, and as such it is most efficiently performed by simulating. And the tool for simulation is the computer, rather than unrealistic hardware. Does a circuit man in today's engineering world build circuits? Rarely—he changes parameter cards. The same thing is true of aircraft: eventually, the designer must build an airplane and prove that it can fly, and even before that he must put the airplane (or parts of it) in the wind tunnel, but the great bulk of his choices are made by analytical, computational, and simulational techniques. Most of our laboratories are fraudulent or spurious in any case. Did anybody ever see anything in the real world that looked like a mass, a spring, and a linear dashpot? No, of course not. Real vibration problems are made of complex interconnections of many masses, many compliances, and many frictional resistances, most of which are nonlinear. The linear approximation is a useful approach to the problem, but it is misleading to tell the student that this approximation is an exact replica of anything in the real world. Why, then, not admit the approximation, and work with the approximation on an analog device, namely an electronic computer? At many places (e. g., at Brooklyn Polytechnic Institute under John Truxal) they have had considerable success with this kind of thing.

In this connection I believe that many laboratory experiments do more harm than good. We give a student a capacitor box which goes inductive at a few thousand cycles, and then we make sure that none of his inputs are above a few hundred cycles. Is this realistic?

On some occasions when I was an undergraduate I wrote up an experiment which I had not done. I wonder if there is anybody in this room who could not or would not make the same statement. I define a good laboratory experiment as one that cannot be written up without having done it. How many of our experiments are of this type? The simuiated laboratory will certainly be less expensive than our present laboratories, and I think in most cases it will be more effective.

Of course, in the case of the laboratory as in all others, there is no reason to go to extremes. We can still supplement the laboratory with occasional real experiments, just as we can supplement with occasional real lectures, with reference to library material, and the like. I do not advocate monomania.

I suspect a number of other people have advocated more or less this type of thing in the past. If it were subsidized by the government, there is the possibility of savings of really large sums of money, in the billions of dollars. If it were done by a book publisher, there would appear to be the possibility of incredibly large profits. It seems to me remarkable that it has not already been done.

PAYNTER: *I want to say I find the controversy delightful. But I feel, and perhaps all of us feel, threatened by part of what you're saying. Because we are physical and we operate by chemical process, I wonder if we couldn't apply a little physical chemistry also, as well as systems engineering, and consider the fact that in a way the lecturer and the teacher play the role of the catalyst, or at least they like to think they do. And the catalyst, oddly enough, is defined as a component which operates in a reaction in such a way that there's no change of mass or energy of the component, and it's my clear understanding that the role of the catalyst in the chemical process industry remains the No. 1 mystery. Would you like to comment on that?*

MACHOL: *It turns out that I took my doctorate in physical chemistry, and I would like very much to comment on that. I would say that this sounds like something you learned in high school because I remember that they told me this in high school and it's just as false as can be. We now, in most cases, understand how a catalyst works. It is not a mystery as far as the chemistry of the thing is concerned. However, I will agree there is some mysterious way in which a professor acts as a catalyst and I thought I had basically commented on this when I discussed the subject of motivation. If you really have a good guy up there, he does this, but I would guess that in 99 per cent of the cases there's very little catalysis taking place.*

REILLY: *In examining the educational process as a communications systems, you've made a very basic assumption and that is that the communication is in one direction only. What about the professor learning from the student?*

MACHOL: *It turns out once again, in our great universities, that we're down to about one course per professor per unit time. The professor then is in contact with his students for about three hours a week and the rest of the time he spends doing all the other things which professors do, which are very much worth doing. There seems to be a feeling that the contact with the student is a comparatively small item. Now I feel that this professor can perhaps be that consultant who is on the other end of the panic button and he will get a much more direct contact with his students. He will, of course, occasionally make one of these canned lecture sets, but he will still be there as an advisor and as the question-answerer, and I think that his contact with the student will be greater than it is today. The professor who gets up in front of a large class of 400 and lectures to them has no feedback to speak of, and this is what the big name, the greatest professors do.*

NEWMARK: *I'd like to ask one question. I'm more concerned about another type of interaction which I don't see as being very convenient on the computer, and that is the interaction between the students themselves. I think what makes some of our great institutions great is the fact that they have an incredibly large number of very excellent students and the students would get along fine if they had much poorer professors than they've got. How do you achieve this with the kind of set-up you're talking about?*

MACHOL: *I don't see any significant difference. The student now sits next to his colleague in the classroom, but hopefully doesn't talk to him during the lecture. In the proposed set-up he will sit next to his colleague in the carrel; he can stop the tape recorder anytime he likes and turn to his colleague and say, "Hey, you, what do you think of the Sox?" or whatever he does say to his colleagues. I see no loss of interaction between students.*

KARNAUGH: *Has anyone proposed instantaneous feedback from the students, whether of the silent-button type or any other, which would, when correlated with the tape of the lecture, allow a certain amount of insight into what was done badly before the lecture might be canned?*

MACHOL: *I would first comment that one of the advantages of the canned lecture is that you can correct it and make it better and better. But I would also like to ask somebody from Urbana to talk about PLATO.*

FENVES: *PLATO is still very much an experimental device. At this stage of the game, we can only run one particular course; of course, it doesn't matter where any individual student is within that course. PLATO has gone through several cycles of evolution. The first one was nothing much more than a programmed learning book, a linear sequence; then they started jumping out of sequence, backing up and so on. Now they are working on several different patterns of teaching and one of them is inquiry teaching. Once this gets implemented, and implemented for the college level, I think we are going to have the kind of interactive environment that Machol was talking about. Namely, there is no lesson plan. It's a pure simulation situation where the student carves out his own path, progresses in any direction that he wishes, can ask questions, back up, or go into side issues, and eventually, hopefully, the program converges and pushes him toward the final result. But there is still a tremendous amount of work to be done, and for a very long time we are going to have to be on-line with a large number of students and constantly update and correct the program so that, while leaving the student quite free to go in any particular direction he wishes, we can produce some kind of convergent pattern so that the objective of the lesson is accomplished.*

PAYNTER: *Answering the question raised specifically, some of Sheridan's people at MIT ran a number of experiments for the better part of a term on instantaneous response and they played both button things, with an indicator in front of the teacher, and they tried lights, they tried meters, they tried everything; one of the most successful things was a sort of joy stick in front of the student. He could manipulatc his own subjective understanding and, when he felt he was slipping out of gear and not following, he would deliberately push the thing over, and, if he felt he was following along with the teacher, he'd push hard over to the other side.*

EMBEDDING A DIGITAL COMPUTER IN A NEW CAMPUS AND SCHOOL OF ENGINEERING

ROBERT M. SAUNDERS
University of California, Irvine
Irvine, Calif.

New universities usually have not started in recent years from "scratch"; typically they have represented expansions of research institutes, specialized instructional activities, or other endeavors. In the University of California two new campuses began instruction in September, 1965, having no constraints on their planning placed by any of the existing physical entities just mentioned.

The Irvine campus is one of these new ventures. Located on one of the largest undeveloped parcels of land (some 88,000 acres) in Southern California on the one hand and the Los Angeles metropolitan area on the other, a unique opportunity has been presented us to build a 27,500-student campus in the midst of, and simultaneous with, the development of a city of 100,000.

An urban university of these dimensions calls for a liberal arts program to provide the background for the successful transformation of students into contributors to society. Thus all students at the University of California, Irvine, will take a minimum of two years' work in the College of Arts and Sciences before going on to more specialized work in that College or in one of the professional schools, of which Engineering is one.

The Irvine campus is a general campus in another sense as well. Along with other campuses of the University we should in due course produce for every 100 B. S. or B. A. degrees, 150 M. S. or M. A. and 25 Ph. D. degrees (100:150:25). At the present time the University of California grants degrees on a 100:186:12 ratio. Nationally comparable ratios are 100:131:5. Thus we shall be deeply involved in the complete spectrum of collegiate education—from freshmen to Ph.D. students. Since our campus must ultimately achieve this complete spectrum, we have started with freshmen to doctoral students.

The School of Engineering is one of the professional schools building upon the two-year liberal arts structure. In our junior-senior and graduate programs we expect to have 500 students in 1970, and an increase by 100 students per year until we reach around 2,500 in 1990. What the upper limit will be is unknown at this time; however, current planning calls for a ceiling of 27,500 for the campus as a whole.

Very early in our planning we noted that digital computation was to become an essential fact of life on the campus as far as scientific computing was concerned. We also realized that we had an opportunity not accorded many campuses to really integrate the computer into the total activities of the campus. We noted that in the university of the future the computer is to play an increasingly important role. Therefore, early in 1964, we made the decision to install a digital computer which would serve the following functions: (1) scientific computation, (2) computer-aided instruction, (3) registration and student record keeping, (4) library search and retrieval, and (5) accounting.

Of these (1) and (5) are well established in most campuses today. As far as record keeping for the Registrar is concerned, (3) is in partial existence on most large and many small campuses today. However, the whole registration procedure, the optimization of student programs, and the development of methods for the utilization of the physical plant are still in formative stages. Similarly library research and retrieval methods (4) need a great deal of engineering to get them off the ground.

Computer-aided instruction (2) is a new area, not well understood, where a tremendous payoff can occur, and we intend to try to bring this about. At this time we really do not know very much about computer-human interaction. Research and evaluation on computer-aided instruction will in time, we are convinced, result in a radically new instructional system.

To begin this endeavor there is now installed at Irvine an IBM 1410/1440/1448 system with card readers and card punches for each of the two processors. We have five tapes and 11 disc packs. The 100,000 character 1410 is the control processor and is directly coupled to the 16,000 character 1440/1448 message processor and switching center. At the present time we have eighteen 1050 terminals connected to the system. Eleven of these terminals are in direct contact with students for at least 12 hours per day and several others are located around the campus in the registrar's office, in class and conference rooms, and in one of the psychological laboratories. Time-sharing operates under a monitor appropriately called "Dean" and runs many routines simultaneously. For student scientific computing we currently run an Irvine version of JOSS (we call it JOSSI). We utilize a course-writer program for developing computer-aided instruction courses.

We also use FORTRAN for faculty and advanced-student research activities. The registrar's activities, experimental accounting procedures, and some library work are also programmed on the computer.

We have a research contract with IBM so that they share in both the cost and the yield in the development of these precedures. Under the terms of this joint endeavor, we provide the real estate, the terminals, and some of the management personnel and programmers; IBM provides some of the equipment and also some programmers. Thus we are engaged in a truly cooperative venture.

To involve Irvine students at the outset, and to provide a basis on which to build computer-aided courses later on, we started a freshman course in digital computing emphasizing algorithm construction. All freshmen in engineering, the social sciences, and the physical sciences are required to take this course. The classes are composed in random fashion of students from the various disciplines, and we do not try to keep special sections for social scientists or for engineers. In my class last fall quarter I had social scientists, physical scientists, engineers, biological scientists, and even a French student. Because the emphasis in the course was on algorithm construction and the breadth of student interest, we were very severely taxed to think of problems we could give students who had had no college mathematics.

Relatively little real instruction is given on programming as such. To de-emphasize programming, we elected to use a student-oriented language. The one adopted was that developed for the Johniac open-shop system at the Rand Corporation, JOSS; this was adapted to the Irvine computer as JOSSI. The feature of this language is that it has immediate accessibility. You simply type your name and identification number and you are in business. JOSSI is easy to learn because it interacts very readily with the students. Its error messages are clear and give clues in plain language as to what is wrong and what should be done. Furthermore, the instruction set is very small, being handled on a single 8½x11-inch sheet of paper.

For large-scale scientific computing we utilize the Western Data Processing Center at UCLA via telephone links and courier.

Our current policy is not to charge anyone for computing time if we can possibly help it, although we do expect those with research funds to pay their share of the processing time. We are toying with the idea of using priority points either in place of, or together with, grants of computing time. Under that system, applicants, no matter whether they were poor or rich, would be given a certain number of priority points. If they elected to use the console between 10:00 and 11:00 in the morning, they might be charged one point per minute. If they elected to use the console or the computer between 3:00 and 4:00 in the morning, they might get 60 minutes for the same point. This method would distribute computer use over the 24 hours of the day.

In general we feel that computer-aided instruction, program debugging, and inquiry on the part of the library, the registrar, or the business office ought to be time-shared from a console, but that scientific-production computing ought to be batch processed, and run as either background or at a non-prime hour.

Several speakers have alluded to the cost issue, but none have really come to grips with the actual costs of a computing facility. Professor Shannon commented on behalf of the Dartmouth installation that they have approximately 50 students per console. This has been very close to our experience, where we find that there are no queues for 40 students per console but there might be if we went to 75. Considering that a terminal costs $1,000 per year, and the central processor from $100,000 to $500,000 a year, for a 10,000-student campus with half of the campus active in computing, $500,000 to $600,000 per year should be available for a computing facility. For a 10,000-student campus this is approximately equal to the library cost per year. Cooper offers encouragement to the academic community when he shows that the American Electric Power computing facility indicates a trend toward saturation in terms of costs while at the same time obtaining an increase in computing output. If this trend as at all predictive, there is perhaps more hope for the campus computer facility.

Presently university computing facilities have three principal sources of funds: (1) university funds, (2) federal grants, and (3) private industry. Grants from government agencies and private industry have been of inestimable help in getting started and in bringing the level of computer activity in the universities to what it is today. What the role of these two outside forces will be in the future is a matter of public and corporate policy. It is not inconceivable that both sources will be curtailed to some extent and it therefore behooves universities to look for internal support. I believe it to be a safe prediction that the percentage support contributed by the universities themselves will increase in the future. If this conclusion is accepted, university operating budgets are in the last resort the place from which the funds will derive. This means a higher student-faculty ratio and less support money per student than is enjoyed today, since most universities, be they private or public, are reaching saturation in the support they can muster per student.

With respect to computer-aided instruction, the School of Engineering will have a strong program. At outset, computer-aided instruction techniques will be applied to laboratory equipment orientation. Should a student come into a laboratory and not be conversant with the operation of a given piece of equipment, he would be asked to address himself to a computer console. It would be his job to learn from the console what he is expected to know about a piece of laboratory equipment before using it. In this fashion we free the instructor for a more Socratic interaction with his students in the laboratory and thereby place the onus for rote instruction on the use of laboratory equipment on the computer.

We also expect to make extensive use of analog computation for undergraduate instruction, and to engage in hybrid-computer research so that an analog computer can be matched with a time-shared digital computer.

We are also preparing engineering design courses in which we will utilize extensively optimization and simulation theory. These courses will be for fourth-year students and will include not only the theory of design but also an actual study carrying the design through to a prototype product.

We also expect to do research on computer-aided design and, in fact, at the present time are so engaged. Current work involves the development of optimization algorithms for systems described by continuous and discrete variable equations of constraint and restraint for linear systems. We expect to expand and extend this study to the nonlinear case.

In conclusion, I would say that we must integrate with the campuses of our universities across the country in the same way libraries have become synonymous with teaching and research. Similarly, we must plan to finance more of our computer facilities out of operating incomes for the campuses, even though this means that we will have less money for equipment and faculty. However, the payoff potential is extremely great since a well-embedded computer system which is designed to be an integral part of a campus does more than provide a facility—it provides an exciting atmosphere and makes for a climate in which we should all be able to produce better students who, in turn, will improve the quality of engineering practice.

ORGANICK: *I'd like to get a little information on the economics of the 7702 transmission. When we looked at the problem of transmitting between Houston and College Station, Texas, which is about 100 miles away, we kept returning to the fact that the bus is cheapest way.*

SAUNDERS: *I don't think there is any way to refute that. It's the turnaround time that we would like to have. You can get 24-hour turnaround time with our bus system. We would like to have four hours turnaround time, which is about all you can buy with that 7700-type system.*

A PLAN FOR IMPROVING INTERDISCIPLINARY COMMUNICATION FOR ENGINEERING DESIGN

BERTRAM HERZOG
The University of Michigan
Ann Arbor, Mich.

INTRODUCTION

Today's session closes a conference emphasizing the impact of computers on education in engineering design. Many speakers have given case histories and examples which make it crystal clear that computers are an aid in design and particularly in teaching design. For the purpose of my remarks, I assume that computers play a useful role. My concern, however, is that as a community of educators we have not taken the impact of computers seriously enough. Nor have we really captured the magnitude of this still-growing impact. Industrial managers suffer from this myopia. My theme is that of improved communication between departments of colleges, between specialists in various disciplines, and between industry and university. This improved interdisciplinary communication must be brought about if we are to meet the challenges implicit in the not-so-distant future activities of engineering education.

The problem is to get a program underway which will meet the needs of the undergraduate student, the engineer employed in industry who is worried about obsolescence, the first-year graduate student, and the faculty research and advanced graduate student. These four classes of people form partially distinct groups which taken together are the ingredients for success of such a program of design education. It is my contention that deliberate and speedy actions must be taken to meet the needs of present and future designers.

There are two factors that contribute to the urgency of the situation. About two decades ago engineering education witnessed the beginning of a change. During the mid-forties a considerable outcry arose regarding the engineer's poor state of preparation to meet the problems of the world and his role in society. Consequently, more non-technical courses were pumped into the curriculum and certain technical courses were discarded. Some deadwood was trimmed; surveying camps were dropped; drawing courses were reduced in number; and other separate courses disappeared which, I am sure, you can name easily. Expediency did not permit reorganization among the topics of importance, although several institutions tried some novel and exciting approaches to the problem.

Then approximately ten years ago, a new onslaught of change faced the engineering curriculum. There commenced a further de-emphasis in the teaching of technology and specialized skills, accompanied by a continuing rise in the teaching of basic knowledge in science and mathematics. Although some of the aforementioned courses again fell victim, this time a large number of courses having design emphasis also got the ax. They deserved it. Design courses in many instances had stagnated and did not match the students' broadening interests which had been aroused by the exploding technologies of the last twenty years. Yet design is an important activity of industrial life. Greater knowledge of the liberal arts is desirable. A sound education in the basic principles is mandatory. Surely all will agree that the changes have been for the better; but now we must bring design back into the picture. The question is: How to do this without expanding the already crowded undergraduate curriculum?

This problem was faced before, when the greater emphasis on science engineering first began. Core courses or curricula were invented by thoughtful teachers who attempted to tie together similar subject matter having different applications within the traditional fields of engineering. Just one such example is embodied in the course in Linear Science and Engineering which is part of the Master-Course concept at John Hopkins University recently described by Professor Grayson.[1] Its content emphasizes the concept of linearity, but examples of applications reflect the particular interest of the student and teacher. Professor Grayson speaks of the success of the approach and also warns of some of the dangers of removing pedagogic repetition. In short, new emphasis need not require throwing out useful and tested material. Before proposing a method for design education, let us consider the other pressures creating the need for a new look at design education.

THE NEED FOR RENEWED INTEREST

The computer revolution, to use the popular phrase, has been accompanied by associated changes in engineering and science. Although high-speed electronic computers were invented to solve engineering or scientific problems in ballistics, they

[1] Grayson, L. C., The Master-Course Concept at The Johns Hopkins University, *Journal of the American Society for Engineering Education*, 56, 5, January 1966.

eventually found their greatest application in the business comunity. However, engineers have found an ever-increasing number of computer applications, which have usually been typified by complicated calculations for the solution of partial differential equations or other problems of advanced mathematical analysis. Even more advanced problems are solved by researchers and new insights are gained with the aid of computers.

Recent developments in computer technology and their applications foreshadow a significantly greater increase in computer use by engineers. I refer to the developments originating at MIT's Lincoln Laboratory and General Motors Research Laboratory. Sketchpad[2] and DAC-I[3] have demonstrated with great clarity the ability to communicate graphic or geometric sketch information to a computer. That graphic communication is an engineer's favorite form of communication need not be debated. Furthermore, computers can be programmed to understand these geometries. Figures and objects can be manipulated, parts can be machined, and other features such as stress distributions may be computed.

Such calculations are done most successfully whenever the designer has the opportunity to interact directly with the computer. This method of working on-line with a machine is not entirely new; in the early days of computing it was quite usual for the user to sit at the computer console and to alter, at his direction, any part of the computation. To do so graphically, however, is new. As machine speeds increased and man's reaction time stayed constant, it became uneconomical to permit this extremely satisfactory communication.

But computer technology again came to the rescue via time-sharing, as demonstrated by Project MAC at MIT and the BASIC System at Dartmouth. It is now possible for many users to communicate simultaneously in an on-line fashion with the machine. The machine shares its power with all of them by capitalizing on any one man's long reaction time (during which he requires no computation) to do the computing chores of the other users. An economical state of affairs is thereby achieved. Those of us who have had the privilege (or is it a right?) of using such a system have found even greater economies in the use of our time. The potential savings in time of technical personnel, our most scarce commodity, is very impressive. Various estimates of the compression ratio of research or problem-solving times range from ten to forty according to some workers. The possibilities in the designer's life will be mentioned later.

For our present discussion, it should be noted that conversational interaction with a computer is exactly what has been needed to solve the design problem and the problem of how design is done.

So we arrive at the state where design has been largely neglected and computer technology has advanced. What about design? What is it? To spend much time on this would be redundant, but a few words of clarification are necessary.

A SIMPLE VIEW OF DESIGN

All writers manage to find their own way of dividing up the subject of design in their quest for a rational description. I believe Professor Asimov's model, given in his monograph "Introduction to Design,"[4] will suit our purpose. A basic knowledge of physics, mathematics, chemistry, or material science is required. This, together with an idea or a specification for a product, initiates a process of synthesis and analysis. This process, in turn, involves assumptions, analysis based upon assumptions, and calculations of results. These calculations produce the first evidence of a trial solution to the design problem. The trial solution is then tested; if it is rejected, new assumptions are made and the design function is repeated until a successful solution is obtained, whereupon the design is complete.

This model of the design process appears to be applicable as much to the design of a part as to the design of a whole assembly. One might also argue that this model is an adequate description of the trial-and-error method of design!

Over the years a mystique has developed around the methods of design. Arguments regarding the relative merits of art versus science in design have raged for years. Please allow me to avoid this argument for it could consume us, but let me acknowledge that both art and science are part of the design process today and will, in all probability, always remain so. This does not mean that we should not seek out the rational methods of design. It also does not mean that one can ignore the rationally unsolvable problems which have been "artistically" resolved for years.[5]

Several significant events have occured to dispel some of the magic of design methods. The Jacquard loom demonstrated our ability to program product

[2] Sutherland, I. E., "Sketchpad: A Man-Machine Graphical Communication System," *AFIPS Conference Proceedings*, Vol. 23, pp. 323-28 (1965).

Johnson, T. E., "Sketchpad III: A Computer Program for Drawing in Three Dimensions," *AFIP Conference Proceedings*, Vol. 23, pp. 347-53 (1963).

[3] Jacks, E. L., "A Laboratory for the Study of Graphical Man-Machine Communications," *AFIPS Conference Proceedings*, Vol. 26, pp. 343-50 (1964).

[4] Asimov, M., *Introduction to Design*, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1962.

[5] Automobiles are built daily at ever faster rates. To find a description of rational methods of analytically based design for automobiles is an unrealistic objective.

variety. Numerical-control machining is now an accepted form of manufacturing for a certain class of goods and its effects are being felt, upstream in the industrial process, at the designer's desk. Numerically controlled drafting methods have been used to speed up some of the design tasks.

The machines have jolted us out of some cherished beliefs about the capabilities of mechanical aids. Now we are about to receive on intellectual jolt. Computer-based methods will permit us to attack design to seek out its rational characteristics.

My apparent obsession with the rational aspects of design, or with design alone at the expense of engineering as a whole, is entirely with purpose. Advances in materials technology and experimental prototype development resulting in new products such as transistors play an important part in engineering and its contribution to our economy. However, our society is not without people devoted to this objective and adequate attention is already given to such developments in our universities. Consequently, special emphasis is needed for the study of design methods.

Our world has many extremely clever designers who over the years have built up an astounding ability to optimize. However, many designers deal with extremely difficult problems by means of comparatively poor rational methods, and thus we have the phenomenon of "little change from last year's model or design." Again, my hat is off to the success of these methods. Today I have very little to offer which might cause good designers to fear their replacement by a computer.

The demands of our economy for technically trained people is tremendous and design organizations feel this shortage. Furthermore, many of our designers must spend deplorably little time being clever and a large amount of time performing clerical or booking tasks.[6]

Many university graduates avoid design activities because they have been bored by dull and unchallenging design courses in school or by equally distasteful activities in industry. In the exploding technology, universities are forced to utilize trite problems to illustrate design situations. Frequently they are limited by the unavailability of real life data. The real world, however, has its problems too; designers in operational situations in existing departments operate daily under such great pressures that they are denied the opportunity to examine the design process from a new point of view, one which might result in increased effectiveness.

The modern designer should have a good understanding of the technology of materials, production methods, and cost to be effective. But he should also know something about problem definition (perhaps system analysis) and optimizing methods. Obviously, universities will not be able to provide him with all these tools for his trade. The technology of manufacturing materials changes so rapidly that it has long been hopeless to give a complete treatment in the undergraduate classroom. Educating engineering students in rational design, that is, optimizing methods with proper cognizance of manufacturing techniques, is a proper activity for the educational institutions.

Appropriate cooperation between industry and university could provide "real-world" data for the classroom. Visualize the industrial engineering student investigating a problem of assembly line balancing; via computer-based communication, he would be permitted access to actual data to test his model. The student would enjoy a much more significant understanding of the problem. Industry would actually receive the benefits of a study not beset by assumptions which void its applicability.

These, then, are the pressures causing the need to initiate a deliberate attack seeking the rational methods in design. New solutions are hard to find, but by harnessing all our available resources in a concerted plan we may have a greater hope for success. It is first necessary to study design methods and to integrate them into an educational process for students and graduate engineers.

The following four ingredients must be considered:

1. Undergraduate education.
2. Continuing education.
3. First-year graduate education.
4. Faculty research and advanced graduate studies.

I will try to separate their respective roles and you will see that I will fail. But do not be alarmed: from this failure we shall draw an extra conclusion.

UNDERGRADUATE EDUCATION

The primary goal of any university is the education of its undergraduate students. Agreement on this point appears to be unanimous. The baccalaureate engineers form the majority of those who practice engineering, particularly in the engineering-design functions. It is well known that young engineers are the useful nuisances who frequently bring about change and innovation in practice. Their training differs sensibly from that of their colleagues, twenty years older.

Many engineering students now learn computer programming as part of their education. While this is essential today to enable any computer utili-

[6] General Motors, in the film presenting their DAC-I System, say that as much as 90 per cent of a designer's time is devoted to these clerical tasks.

zation, it has been difficult to implement such courses and I do not feel that computer programming as such is a proper part of each engineer's education. Using computers, on the other hand, is important. Because of the benefits of computer use, an easier way of communicating problems to the computer and, frequently, an easier way of integrating computer use into the educational process is needed. Several examples of "easy-to-use" computer systems already exist to point the way, but much more work remains to be done in this direction.[7]

The lack of a complete set of special computer programs should not prevent teaching of problem specification, decision theory, optimizing techniques, and the philosophy of rational design. Contributions from research should amplify the repertoire of useful methods and applications to assist undergraduate education.

CONTINUING EDUCATION

Several years will pass before our present or planned efforts in undergraduate education will bear fruit. The current needs of industry and our economy will not wait that long for advanced and new design methods. Since engineering, with its own peculiar needs, has design problems which need solutions. The wrong solutions have cost the taxpayer many millions of dollars. The consumer-goods business seeks ever better refinements in quality, accompanied by improved cost savings to offset competitive pressures. These goals have a direct impact on the design engineer. Consequently, the practicing engineer is not only concerned with his loss of knowledge of technology, but also with his lack of knowledge of modern methods.

Self-education via home reading or night courses has been only moderately successful. The design engineer's daily pressures are tiring and are frequently carried home. Evening courses are thus conducted at the student's worst hours. Top management attitudes have often been neglectful, and consequently the atmosphere for self-education is not always provided.[8]

Re-education has been emphasized for the obsolete executive. Recently one business leader was quoted as stating, "In every company I know of, there are mature drop-outs with college degrees—because they haven't kept up with a changing world, they've cut their career potentials short."[9] This statement certainly can be applied to engineers. Not only have they cut their career potential short, but probably they have hurt their employer's position as well.

Continuing education for engineers in industry is essential. Teaching rational and computer-aided design methods to this group should be a most rewarding challenge. These professionals have had a rea ltaste of the problems of design. To overcome some of their specially tuned prejudices may require special efforts; but, when alerted to other points of view, they will probably be the most enthusiastic proponents.

This educational process will have another special reward. The practicing engineer who engages in additional education is the best source for communicating some of the bothersome details of the actual design world to the academic staff. Without these inputs, the university's role in design education would become most sterile. This is one of the benefits of interdisciplinary communication—that it is a two-way process between the industrial engineer and his university counterpart.

Some serious problems arise in connection with continuing education. My colleague Professor Martin Warshaw[10] in a recent talk here in Chicago addressed himself to these problems. I borrow freely from his remarks. He suggested the notion of a sabbatical leave. The engineer would be paid by his employer while he spends an extended period of time on the campus. The period of time should be at least several months or, preferably, one year. Formal study in quantitative methods for design, balanced with some project or research work to sharpen the effect of the formal studies, would be appropriate. This course of study could lead to a Master's degree, although this should not be the main purpose. An employer's main benefit would be the increased value of his employee. After all, we spend money to renovate machines, why not men?

A stay of one year might be suitable for specialists and middle engineering managers. It is difficult to see how one could gain the attention of senior engineering executives for this length of time. How-

[7] COGO, STRESS and DYANA are typical examples:

Roos, D. and Miller, C. L., "COGO-90: Engineering Users Manual," Department of Civil Engineering, R64-12, MIT Press, April 1964.

Fenves, S. J. et al, "Stress: A Users Manual," MIT Press, 1964.

Theodoroff, T. J., "DYANA-Dynamics Analyzer Programmer Part I, Description and Application," *Proceedings* of the Eastern Joint Computer Conference, pp. 148-51, 1958.

[8] While many companies have generous reimbursement schemes for after-hour education, concurrence of a supervisor is required. Often supervisors only approve those courses *they* believe to be useful on the job.

[9] "Obsolete Executives," The Wall Street Journal, Jan. 24, 1966.

[10] "Education for Physical Distribution Management," a talk delivered by Professor Martin R. Warshaw, Graduate School of Business Administration, The University of Michigan. Seminar on physical distribution management sponsored by the Chicago Association of Commerce, Chicago, Illinois, February 24, 1966.

ever, we must get their attention, for without support from the top there is little hope of implementing an effective specialized program. To obtain this support, a special continuing educational effort must be sustained. This effort, ranging from one-week orientation courses for engineering vice-presidents to one- or two-month courses for other executives and managers, must be a conscious part of the program. These courses need not be relegated to the summer since many universities are leaning toward year-round operation.

THE FIRST-YEAR GRADUATE STUDENT

Graduate study to obtain additional or specialized training is becoming an increasingly important part of the academic output. This third ingredient is likely to offer many fruitful results in the design-oriented education program. Designers will need some additional course work in their areas of specialization, and advanced courses in optimization. Analysis of information systems associated with design are most suitably presented at this level.

Some project experience or research should be an integral part of this training. Research conceivably could be sponsored by industry. The solution of industrial problems might be achieved by teams composed of these students working in conjunction with participants of the continuing education program and under the guidance of members of the faculty. What better arrangement than to have a given industry's employee solve its problem in a different and, hopefully, constructive environment? Probably the earliest useful results for industry would be effected from this aspect of the educational situation, which is also the easiest to implement.

FACULTY RESEARCH AND ADVANCED GRADUATE STUDIES

Research by faculty and advanced graduate students is the fourth ingredient. Results of great significance and lasting impact can be obtained only by the type of effort traditionally associated with research in graduate schools.

Assuming that the other aspects of this educational process are successful and that our industries are staffed with engineers having the new design look, then a whole host of new problems arise. The more rational design becomes, the more readily will its results be integrated with specification writing, manufacturing engineering, purchasing, and all other attendant functions which make the final emergence of the product possible.

Information transmission bcomes a significant problem. Who shall get what information when? Many existing information systems survive only by bootlegging methods and one may seriously wonder what will happen when computers speed the information around. Investigations into the information content of engineering drawings and their role in the industrial process would form a formidable and interesting study.

The role of information and its processing is only one of the many fascinating problems to be solved. Others include stress analysis methods, resource allocation, scheduling, and optimization of multivariate problems. The need for research obviously exists.

IMPLEMENTATION

These four ingredients, undergraduate education, continuing education, first-year graduate studies, and faculty and advanced graduate student research, must be blended together into one functional unit which will devote its main thrust to design education. The demand for modern design education and research will increase. To meet the demand and to exploit the available resources most efficiently requires, in my opinion, the formation of a special activity. For the present purpose I will call this activity the Design Center.

The Design Center is organized for the purpose of identifying the methods of implementing design education at all levels. Furthermore, the Center forms a rallying point for coordinating research efforts and for obtaining support for this research.

Participating faculty and students would still be identified with regular department structures. Degrees earned would satisfy the basic requirements of the department, although a significant part of the student's work would be influenced by the Design Center's staff. This formal affiliation with the usual departments would appear to be less important for the graduate student. Indeed, engineers returning to participate in the continuing education phase, logistically speaking, would be a mixture from many disciplines. Many would be practicing in a field other than the one in which they received their training. The needs of these designers are best met by an interdisciplinary approach. Not only would the Center's design emphasis be attractive, but its interdisciplinary organization would be appropriate.

Having drawn together the faculty from the various departments, one can enhance it even further with recent graduates from institutions where design has been emphasized. Qualified industrial designers might also be attracted to participate in the Center's activities.

Obtaining qualified faculty members is generally a problem and I do not expect it to be any easier for the Design Center. Special efforts must be expended. Whether or not one organizes such a design center, it appears that "major efforts are still required if engineering faculty is expected to use computers creatively and naturally in their teaching."[11] The Center would lend additional emphasis to this need and be able to meet it.

What will the faculty teach? Design is frequently associated with the role of engineering graphics. A freshman-level course emphasizing the information content and the value of the engineering drawing should be part of each engineer's training, provided broader aspects of the engineering profession are also included. In any event, such a course would be part of a design education. During the sophomore year, concentration on experimental and creative aspects of design is appropriate. Upon entering the last half of his education, a student will have completed his training in basic courses in mathematics and engineering sciences and will be thus prepared to pursue advanced studies in a specialty and to integrate these with material emphasizing rational methods of design. Again, proper utilization of existing course material, combined with suitable emphasis on the rational aspects, must be juxtaposed with the importance of the consequent enriched and increased information flow.

Curricula are undergoing change for many reasons and the need for emphasis on design provides yet another. The program can be achieved within the framework of a typical undergraduate program, provided this effort can be combined cooperatively with other curricular revisions.

The graduate and continuing education needs outlined before may be more easily implemented, since graduate curricula are usually less rigidly structured than the undergraduate program. This flexibility could facilitate installation and experimentation with the Design Center activity. After all, graduate programs provide benefits to the faculty, as well as to the student!

The impact of computers has contributed to this renewed emphasis on design. The Design Center must be appropriately equipped with computers or computer service. A large time-sharing system provides the most suitable service environment for the establishment of the Design Center. Assuming that a computer center can supply the major equipment and associated software, then the Design Center need only concern itself with the acquisition of terminal equipment.

The manufacturers gave their reports to this conference a couple of days ago. What we require is a variety of terminals including touch-phones, keyboard terminals, cathode-ray tube terminals for graphics work, and audio-response devices. Curve digitizers, text digitizers, and automatic drafting machines are other useful devices. Film scanners and even more elaborate devices can be added to the list. These suggestions alone have spent the total budgets for many departments for a few years hence. Prices of equipment must and will come down. Technological developments have always looked upon us favorably in this regard.

However, should a pricing miracle occur overnight so that we could purchase all this equipment, it would not do us much good. We would not know how to use it. May I be permitted a rash generalization? Equipment now being produced challenges our ability to use it effectively. The research and development required for efficient computer use for gross design applications must be done soon. Consequently, today's higher prices for equipment must be paid in order to proceed with research. Here again, the Design Center's combined staff can complement each other and use the equipment efficiently.

The supporting staff, out of necessity, would include electronic technicians and equipment operators, programmers and clerical staff. All possible freedom from distracting, although often fascinating, detail work can thus be provided to permit rapid progress on the main thrust toward improved design methods.

Increased costs are unavoidable as we meet the challenges of education for modern society. Many have already suggested that computer services should be reckoned in the same way as library costs. The additional burden of the Design Center should also be figured in this way.

Looking a little further ahead, it would appear that a successful center would attract workers in other disciplines. Business school professors are also concerned with decision-making processes and optimization of business systems. Social scientists, architects, and other professionals could become participants as the problems of environmental design begin to play an increasingly important part in our lives and their professions.

Industry's stake and contributions have been mentioned before. The need and benefits for interdisciplinary communication at all levels is urgent. I believe the proposed scheme offers a mechanism for, at least, a partial solution for engineering design.

CONCLUSION

My remarks have been less than specific with regard to courses and topic outlines. The four ingredients I have chosen to illustrate the problem of design education have not been described in the most precise manner. I indicated earlier that successful separation should not be expected today. Possibly such distinctions are artificial in any case and only served the discussion. The interrelationship of several activities is probably the essential element for success.

[11] Katz, D. L. and Carnahan, B., "The Place of Computers in Engineering Education," *Journal of Engineering Education*, 56, 8, April 1966.

The lack of precision may also be thought to indicate our lack of understanding of the intricacies of the design process—just another reason for starting work on the problem.

There are enough reasons for emphasis and deliberate action. As educators we owe it to ourselves to take a good, new look at design. The need for industry-university cooperation is necessary to success and can be obtained. Let us work together to meet the needs of this problem before the impact of computers beats us about the ears.

SCHMIT: *I'd like to ask a question related to accomplishing interdisciplinary design activities on campus. One of the problems which I have sensed at Case is that it is not entirely successful. It seems to me that interdisciplinary design activities on a campus are handicapped by the lack of pressure of a total system design which is what is forced into interdisciplinary action in industrial situations. I would like to ask if you see any way of artificially creating the pressure that seems to be necessary to bring about genuinely interdisciplinary design in an academic situation?*

HERZOG: *The advantage I experienced in an industrial environment was that I had fifteen people working for me and it was up to my skill to get them working with me; that is, the organization provided the for in this case, and it was up to me to provide the with. I think we characterize universities by the fact that we have a department of so many people working with each other but there is very little staff, pressure, or structure which encourages us to work for each other. Now I'm not suggesting that we invert the organizational structures of universities to this point, but there are certain advantages in an industrial structure that might be advantageously implemented in education. I'm not suggesting that I can organize research this way. We're not getting the output from our many research projects that are being talked about here, or the industrial implementation, into courses with any kind of real meaning. We're getting a lot of duplication, and also isolated instances. In our own school we have this, and I'm sure you do. Since you've made a larger attempt already, I bow to your experience in this regard. I know in a similar sense we have had this problem in industry and one has to work very hard to pull these things together. The tendency to get individual spikes of design done and then thread them together into a system obviously leaves something to be desired in a system analysis approach.*

ZADEH: *At the risk of sounding a perhaps somewhat dissonant note, I must say I was a little surprised when I heard Herzog equate design with computer sciences. Now it seems to me that design, which is the primary function of an engineer, is really the totality of the education that we give our students and so that, in a certain sense, it is much broader than computer science. Computer science is just a small segment of subject area which plays, perhaps, an important role in the training of students for design. This is one point. Another point is that, in the talks that have been given here, it has been shown that computers are more effective at solving certain design problems than classical techniques which do not use computers. But once this stage is over and our students and we ourselves have learned to use computers in that fashion, then the next stage will be really the one that we are not well prepared for and the one we don't really quite understand how to prepare our students for. What bothers me is that it seems that we might get overenthusiastic about what, in reality, constitutes a feeble application of computers, and the stronger applications are the ones that are going to give us trouble in the years ahead.*

HERZOG: *I don't think I equated design to computer science. In contrast, if I may now throw in a little bit more controversy, I will argue strongly, and have for quite some time, that programming courses per se, certainly as programming courses are today, haven't any place in engineering education. That is not the purpose of an engineering education, although people will argue that there are benefits to be obtained in problem definition, problem solution, and so forth, by learning standard types of programming courses. I'm much more concerned with the use of computers in engineering and I do agree that at the present time the only way I can get computers used in engineering is to teach a programming course, but I don't consider that a legitimate objective in the future. There are people who have questioned, in the last couple of days, what the differences are in engineering design and computer science. I don't think we have resolved that, nor shall I try to do so at this time. But I think that what is important in engineering today is to implement the use of computers where necessary in a most painless way, rather than in a most painful way. It is in that sense that I see a lack of involvement, not of individual engineering faculties at the present time, but of engineering schools as a whole, in the question of attacking the problem in design and how to implement computers here. I think we all agree that computers are providing us with a vehicle for looking* *at design problems. All those illusive, intuitive things which we have always tossed under the desk when we were frustrated by not being able to discuss them with each other can now be at least be brought to the fore-front. I suggest to you that implementation is a problem of education, as much as it is a problem for industry. The problems are the same. The computer is going to be the crutch, if you will, that will allow us to do it.*

PAYNTER: *I'd like to pick up from one of the comments Zadeh made earlier having to do with the fact that, as systems become larger, they necessarily tend to become comprehensive and interdisciplinary, and go back and try to suggest for the record some specific answers to Schmit's question. It seems to me that, at the end of the undergraduate curriculum, we are dealing with a student who is still kind of, ipso facto, broadly interdisciplinary. He still is receiving a lot of basic instruction over a very wide range of fields. Often he hasn't even made up his mind that he is going to be a specialist, or what specialization he will follow. I think we have a very real opportunity to present to such a student unstructured problems in which he is forced to use, even in thinking about them, a great many different disciplines. These problems are not two masses and two springs, for instance, but problems where he must distill something out of these. They don't have to be big, because we can rig the stage and present very simple problems that are necessarily comprehensive.*

HERZOG: *No argument here. However, there are some difficulties about large problems which we have had to defer dealing with for so many years because we haven't really been able to tackle them.*

SHANNON: *We produce in engineering education a product which goes out into the environment, and by the environment I mean our whole complex of the United States, and one of the things I think we might consider doing is giving an awful lot of thought to exactly what our educational objectives are and seeing if we can then reach outside the classroom for problems to be brought in and serve as a vehicle. In this regard, one of the things I have been very much convinced of in the last couple of years is that you never tell faculty what to do. You only offer them opportunities to become involved. I think that the first thing any engineer needs is a client; and, if we can select from industry, or from government, certain clients, and offer our staff members the opportunity to become involved in the solution of their problems as a vehicle for the educational objectives that they want to attain, I think that this is one way to start to get this interdisciplinary work as an automatic consequence.*

HERZOG: *It is difficult to argue with the success of your efforts in this regard and I'm glad that you mentioned them. There are, I think, some other problems which are perhaps native to larger institutions of education that are not as evident in the case of the small institution.*

SCHMIT: *Zadeh mentioned that he felt there was an emphasis with regard to computer science and its impact on or the ability to apply it to design. I find this incorrectly exaggerated. I feel that much of the material in the computer science area provides maybe only a small input, but at least a beginning, toward attacking the real design problem. I would like to suggest an idea (which may be controversial), namely, that what we teach in engineering science represents a very small but important underlying ingredient for what is needed for actual engineering analysis when we get out and start looking at a real system. In other words, we don't teach modeling or interpretation, those very vital ingredients of the analysis of real problems, but we do teach the somewhat idealized aspects of engineering science, and these may bear a parallel role with respect to what computer science may have to offer to the whole design problem.*

CONCLUDING REMARKS

N. M. NEWMARK
University of Illinois, Urbana, Ill.

This conference was set up by the Commission on Engineering Education for the purpose of giving whatever assistance it could to faculty at engineering schools who are responsible for courses and curricula in engineering design. Our long-range objectives are to set directions for engineering education as affected by its interaction with computers. We want your suggestions, comments and ideas of ways in which we can help the profession and engineering education reach its long-range objectives, whether this may be in the form of conferences or in terms of other things that we might do. Please give me any suggestions that you might have, or transmit them to Newman Hall, the Executive Director of the Commission on Engineering Education.

I am glad to hear the suggestion that we have another conference. The Commission on Engineering Education is, of course, concerned with where do we go from here and how do we get there, and if enough of you feel that a conference like this is desirable in a year or two, we shall certainly do something about it in the closing discussion.

I would like to summarize my impressions from this conference. There are several factors that I would like to mention concerning, first, the types of use of computers, and second, the topics that we have to be concerned with in the development of our future program. The types of use that have been mentioned which are pertinent are: as a teaching machine; for classroom demonstrations which require input-output equipment, particularly display equipment, and immediate interaction with a computer so that when a problem arises in class you can have it shown on the display; in the solution of problems, which is a sort of traditional use (although we've been doing this for a long while, there are certainly better ways of solving problems); in performing logical operations, which perhaps we have not done as much in the past as we should, and which we are certainly looking at for the future; for such things as filing and information retrieval; and as a tool (although Bert Herzog doesn't like that word) in an integrated systems approach.

To accomplish these objectives we have to be concerned with hardware, especially input-output equipment; software, and this is perhaps our biggest deficiency; and software as it interacts with our staff. We have problems of communication now because of the fact that it usually takes something like a generation to change the background of an engineering faculty. We have to do something to shortcut this. In the past this wasn't such a great deficiency because things didn't change rapidly enough in engineering to make the obsolescence of the faculty such a serious matter. Now it is becoming so serious that we have to give great attention to ways of building up the ability of the faculty to use new techniques, and one of the ways perhaps is to develop such simplified software that *even the faculty* can communicate with the computer!

There is also a very serious problem to which only passing attention has been given so far: that is the matter of financing. Financing of computers is a very difficult problem. A few institutions, because of their small size and good support, or even in spite of the fact that they are relatively large in size but have unusually good support, have been able to solve the problem reasonably well. But there are many schools that cannot afford the kind of equipment and the ways of using this equipment that could really make great inroads into the problem of doing a better job of teaching engineering design. We have to find some solution to this problem.

Finally, there is the most important and most exicting matter of the revision of the curriculum to take advantage of the fact that now there are computers available. The ways in which we have done things traditionally are not the ways in which they should be done to take advantage of the fact that there is a computer. The kinds of things that one learns, the algorithms and so forth, are different for an approach that does not use a computer compared to one that does, and a complete reorganization of the engineering curriculum is one of the things that we must face in the future.

I would like to close by acknowledging the support of the National Science Foundation in helping us present this program, and I would like especially to acknowledge the work that has been done by the steering committee in bringing together this group and arranging for the program and the details of it. That committee consists of Steve Fenves, chairman, Lucien Schmit, Sully Campbell, Frank Branin, Brice Carnahan, and Sam Shapiro. They all have done an excellent job and I would like to express our thanks to them.

I would like also to express our appreciation to the Extension Division of the University of Illinois and especially to the staff of the Chicago Circle campus for their fine cooperation and hospitality.

ATTENDANCE: CONFERENCE ON THE IMPACT OF COMPUTERS ON EDUCATION IN ENGINEERING DESIGN

SECTION G

ANDERSON, ROBERT B.
Carnegie Inst. of Technology
Pittsburgh, Pa.

BALDWIN, GEORGE L.
Bell Telephone Labs.
Murray Hill, N. J.

BAMPTON, MERVYN C. C.
The Boeing Company
Seattle, Wash.

BARKER, CLARK R.
Univ. of Illinois
Urbana, Ill.

BARTHA, TAMAS I.
Pratt Institute
Brooklyn, N. Y.

BAUMANN, DWIGHT M.
Mass. Inst. of Technology
Cambridge, Mass.

BEGG, JOHN
Univ. of South Carolina
Columbia, S. C.

BEILFUSS, CHARLES W.
Meiscon Corporation
Chicago, Ill.

BENNER, RUSSELL E.
Lehigh Univ.
Bethlehem, Pa.

BLATHERWICK, ALLAN
Univ. of Minnesota
Minneapolis, Minn.

BRANIN, FRANKLIN H.
IBM Corp.
Poughkeepsie, N. Y.

BRAUN, LUDWIG
Polytech. Inst. of Brooklyn
Brooklyn, N. Y.

BROWN, R. RODERICK
Honeywell EDP
Waltham, Mass.

CARNAHAN, BRICE
Univ. of Michigan
Ann Arbor, Mich.

CHASEN, S. H.
Lockheed Georgia Co.
Marietta, Ga.

CONWAY, RICHARD
Cornell Univ.
Ithaca, N. Y.

COTTINGHAM, WILLIAM B.
Purdue Univ.
W. Lafayette, Ind.

COWAN, ROBERT A.
Computer Control Co.
Framingham, Mass.

CRANDALL, KEITH C.
Univ. of Washington
Seattle, Wash.

DAVIS, RALPH H.
Westinghouse Electric Corp.
Pittsburgh, Pa.

DAWKINS, GEORGE S.
Univ. of Houston
Houston, Tex.

DI MEO FRANK N.
Drexel Inst. of Technology
Philadelphia, Pa.

DIBOLL, W. B., JR.
Washington Univ.
St. Louis, Mo.

DIETMEYER, DONALD
Univ. of Wisconsin
Madison, Wis.

DIXON, JOHN D.
Univ. of North Dakota
Grand Forks, N. D.

DIX, ROLLIN C.
Ill. Inst. of Technology
Chicago, Ill.

DOUTY, RICHARD T.
Univ. of Missouri
Columbia, Mo.

DYBCZAK, Z. W. PAUL
Tuskegee Inst.
Tuskegee, Ala.

EVERITT, WILLIAM L.
Univ. of Illinois
Urbana, Ill.

FELLING, WILLIAM E.
The Ford Foundation
New York, N. Y.

FENVES, S. J.
Univ. of Illinois
Urbana, Ill.

FORD, JOHN
Univ. of Illinois
Urbana, Ill.

GALL, DONALD A.
Carnegie Inst. of Technology
Pittsburgh, Pa.

GARDNER, NOEL JOHN
Univ. of Western Ontario
London, Ont.

GLASER, ALAN A.
Univ. of Pittsburgh
Pittsburgh, Pa.

GOLDSMITH, C. W.
So. Methodist Univ.
Dallas, Tex.

GOTTFRIED, BYRON S.
Carnegie Inst. of Technology
Pittsburgh, Pa.

GRIFFITH, DEAN E.
Univ. of Texas
Austin, Tex.

HALL, NEWMAN A.
Comm. on Engr. Education
Washington, D. C.

HALL, WILLIAM J.
Univ. of Illinois
Urbana, Ill.

HANCOCK, JOHN
Purdue Univ.
Lafayette, Ind.

HART, DONALD E.
GM Research Lab.
Warren, Mich.

HAUSER, NORBERT
Polytech. Inst. of Brooklyn
Brooklyn, N. Y.

HAWTHORN, QUINTIN J.
Tri-State Coll.
Angola, Ind.

HAYS, ALAN S.
Xerox Corp.
Rochester, N. Y.

HERMAN, HARRY
Newark Coll. of Engr.
Newark, N. J.

HERRING, BRUCE
Auburn Univ.
Auburn, Ala.

HERZOG, BERTRAM
Univ. of Michigan
Ann Arbor, Mich.

HUBER, PAUL E.
Electronic Assoc., Inc.
W. Long Branch, N. Y.

JUHA, MICHAEL
Dartmouth College
Hanover, N. H.

KARNAUGH, MAURICE
Bell Telephone Labs.
Murray Hill, N. J.

KATZ, DONALD L.
Univ. of Michigan
Ann Arbor, Mich.

KIELING, WILLIAM C.
Univ. of Washington
Seattle, Wash.

KINNEN, E.
Univ. of Rochester
Rochester, N. Y.

KREPS, S. I.
Newark Coll. of Engr.
Newark, N. J.

LAWLOR, J. J.
Stevens Inst. of Technology
Hoboken, N. J.

LAWRENCE, WILLIAM N.
Univ. of Michigan
Ann Arbor, Mich.

LEHMANN, JOHN R.
National Science Found.
Washington, D. C.

LEY, B. JAMES
New York Univ.
New York, N. Y.

LICKLIDER, J. C. R.
IBM Corp.
Yorktown Hghts., N. Y.

LING, MARVIN T.
G. E. Computer Equip. Dept.
Phoenix, Ariz.

LONERGAN, WILLIAM R.
RCA Corp.
Cherry Hill, N. J.

LOWE, THOMAS C.
Univ. of Pennsylvania
Philadelphia, Pa.

LUEBBERT, WILLIAM F.
U. S. Military Academy
West Point, N. Y.

MACHOL, ROBERT E.
Dept. of Systems Eng.
Chicago, Ill.

MATTOCK, ALAN H.
Univ. of Washington
Seattle, Wash.

MELIN, JOHN W.
Univ. of Illinois
Urbana, Ill.

MERZ, CHARLES J., JR.
Fairleigh Dickinson Univ.
Teaneck, N. J.

MILLER, CHARLES L.
Mass. Inst. of Technology
Cambridge, Mass.

MISCHKE, CHARLES R.
Iowa State Univ.
Ames, Iowa

MORF, THEODORE
Ill. Div. of Highways
Springfield, Ill.

MOTARD, R. L.
Univ. of Houston
Houston, Tex.

MOWLE, FREDERICK J.
Purdue Univ.
W. Lafayette, Ind.

NEWMARK, NATHAN M.
Univ. of Illinois
Urbana, Ill.

NORTH, WALTER
Univ. of Windsor
Windsor, Ont.

OFFNER, DAVID H.
Univ. of Illinois
Urbana, Ill.

ORGANICK, ELLIOTT
Univ. of Houston
Houston, Tex.

OWEN, C. L.
Ill. Inst. of Technology
Chicago, Ill.

PASKUSZ, G. F.
Univ. of Houston
Houston, Tex.

PAYNTER, HENRY M.
Mass. Inst. of Technology
Cambridge, Mass.

PEARSON, JOHN E.
Univ. of Illinois
Champaign, Ill.

PRYWES, NOAH S.
Univ. of Pennsylvania
Philadelphia, Pa.

REILLY, MATTHEW J.
Carnegie Inst. of Technology
Pittsburgh, Pa.

RESWICK, JAMES
Case Inst. of Technology
Cleveland, Ohio

RIAZ, MAHOUD
Univ. of Minnesota
Minneapolis, Minn.

ROCKHILL, RICHARD A.
Marquette Univ.
Milwaukee, Wis.

ROSENBERG, ARTHUR
Scientific Data Systems
Santa Monica, Calif.

ROSENSTEIN, A. B.
UCLA
Los Angeles, Calif.

ROSENTHAL, PAUL H.
Univac Systems Programming
El Segundo, Calif.

ROSSOW, E. C.
Northwestern Univ.
Evanston, Ill.

SARAJOTL, AMPHORN
Univ. of Colorado
Boulder, Colo.

SAUNDERS, ROBERT M.
Univ. of California
Irvine, Calif.

SCHICK, WILLIAM
Fairleigh Dickinson Univ.
Teaneck, N. J.

SCHILLING, CHARLES H.
U. S. Military Academy
West Point, N. Y.

SCHMIT, LUCIEN
Case Inst. of Technology
Cleveland, Ohio

SEIDER, WARREN D.
Univ. of Michigan
Ann Arbor, Mich.

SEIREG, ALI
Univ. of Wisconsin
Madison, Wis.

SHANNON, P. T.
Dartmouth College
Hanover, N. H.

SHAPIRO, SAMUEL E.
Univ. of Illinois
Chicago, Ill.

SHERIDAN, M. L.
Bucknell Univ.
Lewisburg, Pa.

SIDDALL, JAMES N.
McMaster Univ.
Hamilton, Ont.

SMITH, PAUL
Prairie View College
Prairie View, Tex.

SOOD, GERHARD W.
General Motors Institute
Flint, Mich.

STAGG, GLENN W.
Am. Elect. Power Service Corp.
New York, N. Y.

VIDALE, RICHARD F.
Boston Univ.
Boston, Mass.

WANG, C. K.
Univ. of Wisconsin
Madison, Wis.

WEITER, ELMER J.
Marquette Univ.
Milwaukee, Wis.

WELCH, WALTER R.
Prentice-Hall Inc.
Englewood Cliffs, N. J.

WOLFORD, JAMES C.
Univ. of Nebraska
Lincoln, Nebr.

WOODSON, THOMAS T.
UCLA
Los Angeles, Calif.

WORLEY, W. J.
Univ. of Illinois
Urbana, Ill.

WORTHINGTON, JOHN H.
IBM Corp.
White Plains, N. Y.

ZADEH, L. A.
Univ. of California
Berkeley, Calif.

ZIMMERMAN, JOHN R.
Pennsylvania State Univ.
Univ. Park, Pa.