During the decade beginning in 1962, CUPM made a continuing effort to advise college mathematics departments on curricular matters related to the tremendous growth in the use of the computer and the pervading influence which the computer has come to exert on society. Initial steps in this direction were taken by the Panel on Physical Sciences and Engineering, which issued its Recommendations on the Undergraduate Mathematics Program for Work in Computing* in 1964. Taking account of the significant changes which had recently occurred in the relationship of mathematics to computing and to computing machines, the Panel proposed a program designed to prepare students whose careers were likely to be intimately connected with highspeed computing. The program included reference to three types of courses: (1) mathematics courses of a general nature which should be available for the prospective specialists in computer science; (2) technical courses in computer science; and (3) an introductory course in computer science.

Two years later CUPM commissioned R. W. Hamming of Bell Telephone Laboratories, Inc., to prepare a monograph on Calculus and the Computer Revolution.* Published in 1966, this book describes and illustrates briefly some aspects of computing as they are related to the beginning calculus course.

A task force was appointed in 1966 for the purpose of advising CUPM on a future course of action with regard to computing. The task force suggested the creation of a Panel on Computing, which would work closely with various computing organizations and would have several charges related to the impact of the computer on mathematics education. Such a panel was appointed in 1967.

One of the Panel's projects was to gather and disseminate information regarding the use of computers in introductory calculus courses. A newsletter entitled "Calculus With Computers,"* issued in 1969, contained general observations and summaries of statements from various institutions which had instituted computer-oriented calculus courses.

The Panel's primary aim was to develop a systematic approach to the impact of computers on undergraduate mathematics programs, rather than to address itself to the training of computer scientists per se. (The latter topic had already been considered by the Association for Computing Machinery in its report Curriculum 68--Recommendations for Academic Programs in Computer Science.) The Panel formulated a specific undergraduate program in computational mathematics, combining courses in mathematics, computer science, and computational mathematics--complete with course outlines and suggestions for implementation. This course of study is presented in the 1971 publication

* Not included in this COMPENDIUM

Recommendations for an Undergraduate Program in Computational Mathematics. The main concern of this report is for the education of mathematicians who wish to know how to use and to apply computers.

The report of the Panel on Computing attacked a significant problem: the need for new, innovative courses directly concerned with computational mathematics and computer science. Remaining to be considered, however, was another important question: How should the computer affect traditional mathematics courses? To study this question and related points, CUPM in 1971 appointed a Panel on the Impact of Computing on Mathematics Courses to succeed the Panel on Computing. The new Panel's investigations culminated in the publication of Recommendations on Undergraduate Mathematics Courses Involving Computing in 1972. This document includes outlines for lower-division courses in elementary functions, calculus, discrete mathematics, and linear algebra with stress on algorithms, approximations, model building, and the nature of the entire problem-solving process.

RECOMMENDATIONS FOR AN UNDERGRADUATE PROGRAM IN COMPUTATIONAL MATHEMATICS

A Report of

The Panel on Computing

May 1971

TABLE OF CONTENTS

PREFACE

During the last two decades the development of computers has helped to stimulate the dramatic increase and diversification in the applications of mathematics to other disciplines. In the belief that the time is appropriate for a systematic approach to the impact of computers on undergraduate mathematics programs, the CUPM Panel on Computing presents this report.

> Our basic recommendation is that mathematics departments should experiment with innovative undergraduate mathematics programs which emphasize the constructive and algorithmic aspects of mathematics, and which acquaint students with computers and with the uses of mathematics in computer applications.

A specific undergraduate program in computational mathematics is proposed. This is not a program in computer science, nor is it a minor modification of the traditional undergraduate mathematics major. It is, rather, a program in the mathematical sciences that combines courses in mathematics, computer science, and computational mathematics. It can be used as a basis for further specialization in any of several areas, including computer science, or mathematics, or one of the areas of application of mathematics.

1. Philosophy and Aims of the Program

Since publication of the 1964 CUPM report Recommendations on the Undergraduate Mathematics Program for Work in Computing, computer science has developed as a separate area of study. More and more colleges and universities are establishing computer science departments, and the number of students enrolled in computer science programs is increasing rapidly. The need for separate curriculum studies in this new area was recognized by the Association for Computing Machinery, and in 1968 its Curriculum Committee on Computer Science published a report entitled Curriculum 68--Recommendations for Academic Programs in Computer Science. This widely acclaimed report is still regarded as giving a good description of curricula in computer science. Its recommended minimal mathematics preparation is about equivalent to that usually required of students in the physical sciences and engineering.

More recently, three trends have become noticeable. First, there appears to have developed a strong tendency on the part of computer science programs to minimize prerequisite requirements in traditional mathematics, particularly analysis, and also to underemphasize or even to disregard most areas of scientific computing.

Second, many disciplines, including in particular the biological, social, and behavioral sciences, have become increasingly mathematical, giving rise to a need in these fields for expanded education in mathematics and in scientific computing. Finally, the computer has begun to have a direct effect upon mathematics courses themselves. New courses, particularly in computationally-oriented applied mathematics, are being introduced into many mathematics curricula, and traditional courses are being modified and taught with a computer orientation. As an example of the latter we cite only the teaching of calculus. Approximately 100 schools now offer a course in calculus using the text Calculus, A Computer Oriented Presentation, published by the Center for Research in College Instruction in Science and Mathematics. Other computer calculus projects were reported in the 1969 CUPM Newsletter, "Calculus with Computers," now out of print.

These three trends all indicate a need for the mathematics community to accept a responsibility for mathematical or scientific computing and to broaden educational opportunities toward a more encompassing "mathematical science" in which students may explore the areas of overlap between pure and computational mathematics, as well as computer science. There is thus a need for innovative undergraduate programs which provide for a wide range of options, different opportunities for graduate study, and a variety of future careers.

A new view of mathematics as a mathematical science in the above sense raises many curricular questions, to which several CUPM panels have begun to address themselves. In particular, a need arose for reappraisal of the already-cited 1964 report. Such a reappraisal is desirable if for no other reason than that a large number of all undergraduate mathematics majors are likely to find themselves later in some computer-related field.

The present report is the result of such a reappraisal by the CUPM Panel on Computing. From the outset it was evident that the aims of this report should be different from those of the earlier work, since its intended audience is different. The present report does not address itself to the training of computer scientists. Instead, its concern is for the education of mathematicians who will know how to use and to apply computers. Programs in computational mathematics necessarily have different objectives than do programs in computer science.

In accordance with our previous remarks, the mathematics program presented here is intended to be a departure from the traditional undergraduate mathematics curriculum. It should not be regarded, however, as a replacement for that curriculum, but rather, together with it, as one of several equally valid options for students of the mathematical sciences. It should meet the needs of students who plan to enter careers in scientific computing or who wish to enroll in graduate programs in computationally-oriented applied mathematics. With some suitably selected options during the senior year, a continuation in many computer science graduate programs should be possible. With other options, a continuation in pure mathematics

should also be possible. At the same time, several of the courses included in the program meet the mathematical needs of students in other disciplines and may also be appropriate for prospective secondary school mathematics teachers.

The program proposed here is presented in a spirit of open experimentation, not as a final product. In its design the Panel has been neither as conservative nor as radical as it might have been. For instance, a conservative approach might be to combine a list of suitable mathematics courses of a traditional nature with a complementary list of computer science courses. This is easily accomplished in an institution having both a mathematics and a computer science department, but it leads to a large number of required courses and provides for little or no interaction between the two parts of the program. At the other extreme stands a curriculum in which computing has been completely integrated with the mathematical material, either by the introduction of new courses or by the repackaging of old ones.

In designing its program the Panel has taken a path somewhere between the extremes indicated above. Several new computer-oriented mathematics courses are described here; at the same time, some standard computer science and mathematics courses are included and, in particular, no recommendations are made concerning the redesigning of standard mathematics courses, such as the calculus, to include computer use. Where they are available, such computationally-oriented basic mathematics courses could be ideal components of this program, but their definition still requires considerable study and experimentation. The Panel felt that such a study on its part would serve only to divert its attention from its main concern, namely, the description of a new curriculum in computational mathematics for the undergraduate mathematics major which can be implemented in many institutions without excessive cost or delay.

In this latter connection the Panel believes that its program can be offered even by smaller colleges having suitable access to educational computing equipment, with only modest additions to their mathematics staffs. More specifically, through the junior year, the new computationally-oriented mathematics courses recommended here number only four. These, together with the three basic and relatively standard computer science courses, could be handled by the equivalent of one mathematician interested in applied mathematics with an emphasis on computing and numerical analysis and one specialist in computer science. The remaining core courses can be taught by the other members of the mathematics department. Clearly, this small staff could offer only a few of the additional courses listed in this report as possible electives, but the Panel believes that even such a minimal program would be desirable for many students.

2. Recommendations and Brief Course Descriptions

For a major undergraduate program in Computational Mathematics we recommend a basic core curriculum of 12 one-semester courses: five in mathematics, four in computational mathematics, and three in computer science. We will refer to these courses in the sequel, respectively, by the symbols M1, M2, M3, M4, M5, CM1, CM2, CM3, CM4, C1, C2, and C3.

Each of the courses carries 3 credits; at the same time it is desirable that some of the computer-oriented courses include a scheduled laboratory period for which additional credit may be awarded. As described below, this sequence can be handled in three years, leaving the senior year for electives, also set forth below.

2.1 Basic Component

Before describing the 12 courses in the Basic Component, it may be instructive to illustrate one way of imbedding them into the first three undergraduate years. In the chart on page 534, arrows indicate the "prerequisite structure," i.e., the dependency of each course on those which precede it. Notice that two courses are recommended for each semester. Mathematical progress within the program is not different from that in standard programs. If the student wishes to switch to pure mathematics after sampling the eight core courses of the first two years, it will be a simple matter for him to do so with no loss of mathematical pace. It should also be noted that of the CM and C courses, three are taught in the first semester and four in the second semester of each year. This part of the program could easily be handled by the equivalent of two teachers in a small college where multiple sections are unlikely.

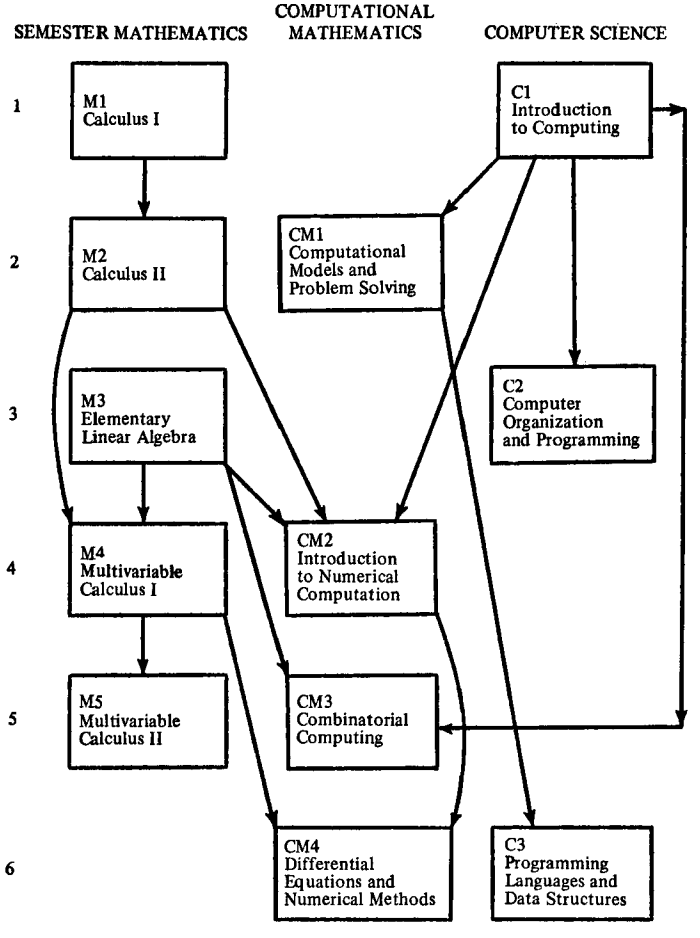
Let us now describe these 12 courses briefly, leaving detailed course outlines and references for Section 4.

a) Mathematics courses

These five courses are described in the CUPM document Commentary on A General Curriculum in Mathematics for Colleges, page Incidentally, the committee which produced the Commentary has already noted that there is little need for M3 to require M1-M2 as explicit prerequisites. This fact has been observed in the chart.

M1	Calculus I
M2	Calculus II
M3	Elementary Linear Algebra
M4	Multivariable Calculus I
M5	Multivariable Calculus II

CHART SHOWING ONE WAY OF IMBEDDING THE BASIC COMPONENT INTO THE FIRST THREE UNDERGRADUATE YEARS. ARROWS INDICATE THE PREREQUISITE STRUCTURE.



b) Computational Mathematics

These courses constitute the heart of our program. While their spirit is mathematical, computing plays an important role in each. The courses CM1 and CM3 are novel in character, while CM2 and CM4 are intended to replace the traditional first courses in Numerical Analysis and Ordinary Differential Equations. In the initial phase of implementing this program the traditional versions of these courses could be used temporarily in place of CM2 and CM4, thereby allowing the faculty to concentrate first on the development of the new courses CM1 and CM3.

CM1. Computational Models and Problem Solving

Prerequisite: C1

The purpose of this course is to introduce students early in their programs to a wide variety of different computer applications. This is to be accomplished mainly through the construction and interpretation of computational models for several interesting and worthwhile practical problems from various disciplines, including the biological and behavioral sciences as well as the physical sciences and mathematics.

The spirit in which the course is presented is of utmost importance. The applications discussed in the course should be reasonably realistic and comprehensive, and the students should become aware of the very serious difficulties and limitations that can arise. Questions should be raised about the validity of models, the effect of numerical errors, the significance of statistical results, the need for data verification, the difficulties in testing programs, documentation, etc. Whenever possible, the basic mathematical aspects of the different models should be discussed in general and related to the computational results. However, since the course is intended for freshmen or sophomores, no attempt can be made to enter into any deeper analysis of specific mathematical questions. With a proper balance between the computational and mathematical points of view, the course should provide the students not only with an appreciation of both the potential and limitations of computer applications but also with an interest in learning more about the many relevant areas of mathematics.

The outline included in Section 4 places special emphasis on the use of computational models for the simulation of random and nonrandom processes, although a few numerical and nonnumerical computer applications are also included. The latter types of problem will be considered in more detail in the subsequent courses CM2 and CM3.

It should be noted that this course may also be of considerable value and interest to students outside the present program.

CM2. Introduction to Numerical Computation

Prerequisites: C1, M2, M3

This first course in numerical analysis may be taken in the sophomore year. Since it is based on as little as one year of analysis, the emphasis should be more on intuition, experimentation, and error assessment than on rigor. The methods considered should be amply motivated by realistic problems. It is better to treat a few algorithms thoroughly than to be exhaustive in the number of algorithms considered. Students should be expected to program and run a number of problems on a computer, and considerable time should be spent analyzing the results of such runs. In particular, the analysis of roundoff and discretization errors, as well as the efficiency of algorithms, should be stressed.

Topics should include the solution of linear systems, the solution of a single nonlinear equation, interpolation and approximation (including least squares approximation), differentiation and integration, and elements of the numerical solution of eigenvalue problems.

CM3. Combinatorial Computing

Prerequisites: C1 and M3

Combinatorial computing is concerned with the problem of how to carry out computations with discrete mathematical structures. It bears to combinatorial (discrete, finite) mathematics the same relationship that numerical analysis bears to analysis. Numerical analysis is much more widely known and much better developed than combinatorial computing. However, there are many reasons to believe that within the next decade combinatorial computing will rival numerical analysis in its importance to computer users. In fact, outside of the traditional areas of applications of mathematics to the physical sciences, discrete mathematical structures may occur more frequently than continuous ones, and even in large problems in the physical sciences data-handling considerations lead quickly to questions in combinatorial computing.

This course is intended as an introduction to the emerging field of combinatorial computing. Its objectives are (1) to acquaint students with certain types of problems which occur frequently when problems are formulated in combinatorial terms, so that they are able to recognize them when they encounter them in disguise, and (2) to teach students certain important concepts and proven techniques which experience has shown to be useful in solving many combinatorial problems, particularly on a computer.

Typical topics to be covered in the course are the representation of integers, sets, and graphs; counting and enumeration techniques; sorting and searching methods; and selected problems and

algorithms in graph theory. Students should be expected to write programs for various algorithms and to experiment with their application to appropriate problems.

CM4. Differential Equations and Numerical Methods

Prerequisites: CM2 and M4

This course is intended to replace the more traditionally oriented course in differential equations in which the focus is often on nonconstructive developments. It has the objective of introducing the student to key concepts underlying the qualitative understanding of differential equations as well as to methods for constructing their approximate solutions. It is intended for the junior year. The historical development of the subject is closely related to the physical and engineering sciences; nevertheless, it is recommended that examples from biology, economics, and other fields be chosen where possible, so as to draw upon a student's intuitive understanding of the processes illustrated. Some further suggestions for such material can be found in the CUPM report Applied Mathematics in the Undergraduate Curriculum, page 705.

As a result of this course the student should have confidence in his ability to develop an approximate solution of a differential equation, be able to discuss the basic qualitative behavior of the solution, and have an appreciation of the importance of analytic methods in furthering his understanding of the subject.

Typical topics should include a discussion of simple linear equations, the initial value problem for the first-order equation $y' = f(x,y)$ and some methods for its numerical solution, a basic introduction to first-order systems and their applications including plane autonomous systems, and finally some topics relating to boundary value problems.

c) Computer Science

The following three courses represent certain modifications of several of the basic courses in Curriculum 68. All three courses should not consist simply of lectures but should also incorporate a scheduled laboratory period.

C1. Introduction to Computing

Prerequisite: College admission

This first course in computing has by now become standard in many institutions. The 1964 CUPM report Recommendations on the Undergraduate Mathematics Program for Work in Computing recommended a particular version of this course, and the corresponding course B1 in

Curriculum 68 has been widely referenced. The course serves several purposes:

(1) To develop an understanding of the concept of an algorithm and of the algorithmic formulation of methods for the solution of problems on a computer.

(2) To train the student in the use of at least one algorithmic programming language and to introduce him to the basic structural aspects of such languages.

(3) To acquaint the student with the basic characteristics and properties of computers.

For the program proposed here the stress of the course should be on problem solving by computer. Accordingly, the student should be assigned a number of different problems both of the numerical and nonnumerical type, including at least one larger project.

C2. Computer Organization and Programming

Prerequisite: C1

The purpose of this course is to provide the student with a basic introduction to the structure and organization of digital computers and to the use of assembly language programming systems, without becoming involved in a too-detailed discussion of computer hardware or assembly language programming.

The course proposed here is in part similar to the course B2 in Curriculum 68 with the addition of some topics from the course I3 in the same report. However, unlike those courses, it has primarily a survey character. Typical topics include computer structure, assembly languages, data representation, addressing techniques, elements of logic design, discussion of the principal units of a digital computer, systems software, and a survey of contemporary computers.

C3. Programming Languages and Data Structures

Prerequisite: CM1

This course is intended to introduce the student to some of the elements of programming languages as well as to certain important techniques of organizing and linking together information stored in a computer. Topics covered in the course include the basic structure of algorithmic languages, tree and list structures in a computer, string manipulation, data structure and storage allocation, and basic aspects of languages and grammars. The students should become acquainted with at least two different-level languages, such as a string manipulation language and an advanced algorithmic language.

The course covers a number of topics from the ACM courses I1 and I2 but is otherwise novel in character. Some instructors may find it desirable to use CM3 as a prerequisite; this would be similar in spirit to the approach of the ACM recommendations. But it is equally conceivable to introduce C3 as a prerequisite for CM3, allowing a much wider range of computational assignments in the latter course.

2.2 Elective Component

Given the Basic Component described above, and depending on the student's particular interests, there are several ways to round out a good major program. Broadly speaking, possible technical electives can be grouped under the following six--somewhat overlapping--categories, not necessarily in order of importance:

a) Mathematics
b) Probability and statistics
c) Computationally-oriented mathematics
d) Other applied mathematics
e) Computer science
f) Other disciplines

The specific courses listed here under each of these headings are not meant to exhaust all possibilities; clearly, there are various other choices and variations. If the Basic Component of the program has been completed during the first three years, the elective courses will--most probably--be concentrated during the senior and part of the junior year. But other arrangements of the Basic Component are also possible, thereby allowing for a distribution of elective courses throughout most of the undergraduate program.

a) Mathematics

Several of the courses offered as part of the standard mathematics curriculum can serve as electives for a computational mathematics program. This involves, in particular,

Introductory Real Variable Theory (Mathematics 11-12 of GCMC)
Complex Analysis (Mathematics 13)
Introductory Modern Algebra (Mathematics 6M)
Linear Algebra (Mathematics 6L)
Introduction to Mathematical Logic

The Basic Component, augmented by a year course in real variables and a year course in algebra, would constitute minimally adequate preparation for graduate study in mathematics. These additions could easily be achieved in the senior year.

The standard introductory course in ordinary differential equations has not been mentioned here again since it was replaced by CM4.

A beginning course in partial differential equations is included in subsection c) below.

b) Probability and Statistics

Statistical computations represent a large percentage of scientific computing work in many disciplines. Accordingly, the Panel believes that a good introduction to probability and statistics is highly important to students in a program of the kind discussed here. In fact, it may be very desirable to require such an introduction of all students in the program.

The Panel recommends a one-year combination of probability and statistics with M4 as a prerequisite. The first semester should provide an introduction to probability, with the second covering suitable topics from statistics. Courses like this are already offered in many schools, and recommendations about the material to be covered have been set forth by the CUPM Panel on Statistics in Preparation for Graduate Work in Statistics, page 459.

For the purposes of a computational mathematics program it may be highly desirable to integrate computational aspects directly into these courses. But in line with the approach taken in this report, the Panel did not wish to make any such specific recommendations at this time.

c) Computationally-oriented Mathematics

The courses grouped under this subheading are similar to CM1-CM4; that is, their spirit and content are mathematical, but computing plays an important role in each. Accordingly, it is most desirable that a program in computational mathematics include at least some additional courses of this nature.

From among the variety of possible topics the Panel decided to select five course areas which appear to be fairly representative.

Numerical linear algebra

This course covers the description and analysis of some of the principal computational methods in linear algebra. It uses CM2 and M3 as prerequisites and could replace the standard advanced linear algebra course for students in this program. Typical topics might include a thorough discussion of elimination methods and of Wilkinson's backward error analysis, iterative methods for large linear systems and the corresponding basic convergence results, and methods for solving eigenvalue-eigenvector problems. The various topics should be motivated and illustrated by means of different applications.

Courses like this have become almost standard in many institutions. The course material can be found, for example, in parts of

the following texts:

Forsythe, George E. and Moler, Cleve B. Computer Solution of Linear Algebraic Systems. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1967.

Householder, Alston S. The Theory of Matrices in Numerical Analysis. Waltham, Massachusetts, Blaisdell Publishing Company, Inc., 1964.

Noble, Ben. Applied Linear Algebra. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1969.

Varga, Richard S. Matrix Iterative Analysis. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1962.

Wilkinson, James H. The Algebraic Eigenvalue Problem. New York, Oxford University Press, Inc., 1965.

Applied modern algebra

The purpose of this course is to introduce the student to the discrete algebraic structures most commonly used in applications. It is intended to replace the standard modern algebra course (Mathematics 6M of GCMC) for those students who are concerned with applications of algebra rather than with algebra as pure mathematics. Whereas the topics are in general not intended to be treated in depth, the treatment should be adequate enough in each case to enable the student to read independently in more complete expositions. Accordingly, the presentation should include formal definitions and proofs of fundamental theorems, but at the same time there should be considerable emphasis on practical applications.

While courses on applied and computational linear algebra have become reasonably common, the same cannot be said about courses on applied modern algebra. Moreover, at present there exists essentially only one text on this topic, namely,

Birkhoff, Garrett and Bartee, Thomas C. Modern Applied Algebra. New York, McGraw-Hill Book Company, 1970.

This book contains material for a full year course. A one-semester course on the senior level with a prerequisite of CM3 might begin with a review of set algebra and an introduction to semigroups and groups and some of their applications. Then the stress could be placed on partially ordered sets, lattices and Boolean algebra, and their applications in switching algebra and logic. Another approach would be to play down Boolean algebra and to stress rings and fields, including, in particular, polynomial rings and finite fields, and their applications to coding theory.

Optimization

Optimization problems arise frequently in scientific computer applications. This includes problems from the entire area of mathematical programming as well as from optimal control theory, calculus of variations, and from parts of combinatorics. A one-semester introductory course in optimization problems, with CM2 and M4 as prerequisites, is therefore a highly desirable elective in a program of this kind.

Such a course--not stressing computational aspects--was described in the CUPM report Mathematical Engineering--A Five Year Program, page 649.* It begins with a discussion of specific examples of typical optimization problems from the various cited fields, and continues with an introduction to convexity and n-space geometry, Lagrange multipliers and duality, and the Simplex method. Then it turns to some combinatorial problems and to elements of the classical calculus of variations and of control theory. In a more computationally-oriented version of the course it appears to be desirable to delete the latter three topics and to present instead an extended coverage of the numerical aspects of linear programming, as well as a discussion of transportation problems. The course could then end with an introduction to numerical methods for convex programming problems. The student would be assigned computational projects involving some of the many available library subroutines; in fact, an important by-product of the course in this form might be to familiarize the students with the extensive computational effort that has already been spent in connection with mathematical programming techniques.

There is an extensive list of available references relating to this course. Without attempting to be comprehensive, we mention only the following books:

Berge, Claude and Ghouila-Houri, A. Programming, Games and Transportation Networks. New York, John Wiley and Sons, Inc., 1965.

Dantzig, George B. Linear Programming and Extensions. Princeton, New Jersey, Princeton University Press, 1963.

Hadley, George F. Linear Programming. Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1962.

Hadley, George F. Nonlinear and Dynamic Programming. Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1964.

* See also the CUPM reports Recommendations on the Undergraduate Mathematics Program for Engineers and Physicists [page 628] and Applied Mathematics in the Undergraduate Curriculum [page 705].

Künzi, Hans P.; Tzschach, H.; Zehnder, C. Numerical Methods of Mathematical Optimization with ALGOL and FORTRAN Programs. New York, Academic Press, Inc., 1968.

Polak, E. Computational Methods in Optimization. New York, Academic Press, Inc., 1971.

Partial differential equations and numerical methods

The general aim of this course is to survey the standard types of partial differential equations, including, for each type, a discussion of the basic theory, examples of applications, classical techniques of solution with remarks about their numerical aspects, and finite difference methods. By necessity, most proofs of existence and uniqueness theorems and of the properties of the numerical methods are to be omitted.

A course of this kind--based on CM4 and M5--requires in general two semesters, and even then it will be very demanding of the students at the senior level. Typical topics include first-order equations and the elements of the theory of characteristics for linear and quasilinear equations; linear second-order equations in two variables; classification; canonical forms; a discussion of the wave, diffusion, and Laplace equations; and a survey of some topics about other equations. For a description of a one-year course on partial differential equations--not stressing numerical methods--see also the CUPM report Mathematical Engineering--A Five Year Program, page 649.

There do not appear to be any entirely appropriate texts for this course. The following are some possible titles:

Ames, William F. Numerical Methods for Partial Differential Equations. New York, Barnes and Noble, 1970.
Probably too difficult as a text for a first undergraduate course, but valuable as a reference for the course.

Berg, Paul W. and McGregor, James L. Elementary Partial Differential Equations. San Francisco, California, Holden-Day, Inc., 1966.
Elementary introductory text, but does not emphasize numerical methods.

Forsythe, George E. and Wasow, Woolfgang R. Finite Difference Methods for Partial Differential Equations. New York, John Wiley and Sons, Inc., 1960.
Important reference for numerical methods.

Mitchell, A. R. Computational Methods in Partial Differential Equations. New York, John Wiley and Sons, Inc., 1969.

Weinberger, Hans F. A First Course in Partial Differential Equations. Waltham, Massachusetts, Blaisdell Publishing Company, 1965.
Introductory text which places special consideration on physical applications.

Introduction to applied functional analysis

The purpose of this course is to present some of the basic material of elementary functional analysis as it is of use and importance in numerical and applied mathematics. With a prerequisite of CM2 and M5, the course includes an introduction to metric spaces, the contraction mapping theorem and various of its applications, normed linear spaces, linear and nonlinear operators, the differential calculus on normed spaces, applications to iterative processes such as Newton's method, minimization techniques for nonlinear functionals on Banach spaces, and, if time permits, some discussion of the relationships between functional analysis and approximation theory.

By necessity, the material has to be presented from a geometrical and intuitive viewpoint rather than in a formal and abstract manner. Some of the results should be explored further by applying them to specific computational problems; here team projects may be very appropriate.

The following are some texts which cover parts of the material mentioned above:

Collatz, Lothar. Functional Analysis and Numerical Mathematics. New York, Academic Press, Inc., 1966.
Survey of many of the interactions between the two fields.

Davis, Philip J. Interpolation and Approximation. Waltham, Massachusetts, Blaisdell Publishing Company, 1963.
For the connections to approximation theory.

Dieudonné, Jean. Foundations of Modern Analysis. New York, Academic Press, Inc., 1969.
For the differential calculus on normed linear spaces.

Goffman, Casper and Pedrick, George. First Course in Functional Analysis. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1965.
For much of the basic material; not numerically oriented.

Goldstein, Allen A. Constructive Real Analysis. New York, Harper and Row, Publishers, 1967.
For minimization methods.

Kantorovich, L. V. and Akilov, G. P. Functional Analysis in Normed Spaces. Elmsford, New York, Pergamon Press, Inc., 1964.
Contains a detailed discussion of Newton's method.

Kolmogoroff, A. N. and Fomin, S. V. Elements of the Theory of Functions and Functional Analysis, vol. I. Baltimore, Maryland, Graylock Press, 1957.

Schechter, Martin. Principles of Functional Analysis. New York Academic Press, Inc., 1971.

In some of these courses it may be desirable to add a second semester in order to provide a more extended coverage of the material. This applies also to CM4, where a second semester is probably very desirable for many students.

The Panel believes that students in a program such as this might benefit by being able to deepen their knowledge in graph theory and combinatorics beyond the material covered in CM3. A course in this area is described in Applied Mathematics in the Undergraduate Curriculum, page 734.

d) Other Applied Mathematics Courses

As discussed in the beginning, a main aim of this program is to provide the student with a basic understanding of the application and use of computers in the solution of scientific problems. Accordingly, it will be most important that the student acquire a certain familiarity with at least some of the many applications of mathematics and with mathematical model building.

A number of suitable topics for an applied mathematics course are discussed in Applied Mathematics in the Undergraduate Curriculum, page 705 . Outlines for some courses in physical mathematics are described in the CUPM report Mathematical Engineering--A Five-Year Program, page 649. From this latter report we mention, in particular, the following courses:

ME3	Mechanics
ME8	Electromagnetics
ME9	Thermodynamics and Statistical Mechanics
OR2	Operations Research
OR3	Systems Simulation
OM3	Celestial Mechanics
OM4	Orbit Theory
CT2	Control
CT4	Linear Systems
CT7	Information Theory

It should be stressed that for the purposes of this program the specific topics covered in any of these courses are not as important as the applied mathematical spirit, that is, the emphasis on model building, on analysis of the model, and on interpretation of the results.

It should also be noted that by listing these courses separately from those in the previous subsection we do not mean to imply that little or no computational work is to be involved here. In fact, in many of these courses computer applications might prove to be of considerable value and might strengthen the student's understanding of the interrelationship among scientific problems, mathematical models for them, and numerical methods for finding approximate solutions of these models.

e) Computer Science

In an institution with an ongoing computer science program, many of the courses offered as part of that program can serve well as possible technical electives in this curriculum. We mention, in particular, the following courses described in Curriculum 68:

I4	Systems Programming
I5	Compiler Construction
I6	Switching Theory
I7	Sequential Machines
A1	Formal Languages and Syntactic Analysis
A2	Advanced Computer Organization
A4	System Simulation
A5	Information Organization and Retrieval
A7	Theory of Computability

Some introductory courses in mathematical logic covering topics from I7, A1, and A7 are also offered by many mathematics departments and may serve as possible electives.

Finally, it may be of interest to note that with the addition of three or four courses, such as I4 and I5 or A1, to the Basic Component, a student would meet more than the minimal requirements in Curriculum 68 for an undergraduate major in computer science. Such additions could easily be achieved in the senior year.

f) Other Disciplines

This last and yet by no means least important subgroup of possible electives concerns courses in any of the disciplines outside of mathematics which are sources of mathematical computing problems. The Panel firmly believes that an understanding of the ideas, principles, and methods of at least one such area is a basic ingredient of the education of a computational mathematician and hence that any student in this program should take at least some suitable courses in another discipline. It should be stressed that this need not be the traditional introductory physics sequence, but that beginning courses in the engineering, biological, behavioral, or social sciences might be equally appropriate. The specific type and number of courses depends in each case on what is available, the field selected, and the student's depth of interest.

3. Implementation of the Program

3.1 Staff

It was stated earlier that the program of the Basic Component could be carried out by a mathematics department with the equivalent of one faculty member interested in numerical analysis and computing, and one in computer science. Such a department could offer some of the elective courses as well, depending on the interests of its members. A year course in Probability and Statistics is already taught in many colleges, and courses in Modern Applied Algebra are beginning to appear in addition to, or as replacements for, the usual courses in Abstract Algebra. Courses resembling those in Optimization or Applied Functional Analysis are also offered by many colleges.

Thus, while the entire program could not be offered except in an institution with several faculty members in applied and numerical mathematics as well as in computer science, much of it--and especially the Basic Component--may be possible in a small college with an expanded mathematics department as described above, provided a computer is available.

This leaves, of course, the question of staffing the computing facility itself, which in turn depends strongly on the nature of that facility. In most cases, such a facility requires the supervision of at least one professional manager or director, who in turn may be capable of teaching the necessary computer science courses in this program. Besides this person, many colleges have found that the problem of staffing the computing laboratory can be solved in part, or even completely, through the students themselves. One of the virtues of the computer as an instructional device is the personal involvement that it demands of and readily receives from the students. They learn quickly for the most part and teach one another very effectively. They serve well in many jobs associated with the operation of the computer facility. To bring them formally into the teaching process is sensible and rewarding.

3.2 Facilities

Apart from dealing with the arrays of desk calculators which have served statistical laboratories in the past, mathematics departments have not faced the wide variety of problems connected with the incorporation of laboratory work into their academic programs. The implementation of this program necessarily requires careful planning and maintaining of proper laboratory facilities. Because of sustained increases in costs of education, college administrations are understandably hesitant to incur major new expenditures. The following discussion is directed toward helping to clarify or distinguish among various factors which might characterize a computational facility suitable to this program.

The principal ways of incorporating computer use into an educational program can be characterized as follows:

1) Discussion of computational results obtained directly or indirectly by the instructor.

2) Student use of computers outside the classroom in a batch mode. Here, typically, programs are collected and submitted to be run together on a computer, without the possibility of further interaction from the originator. Very often the input is in the form of punched cards.

3) Student use of computers outside the classroom in a time-sharing mode. Here either simple teletypewriters or more elaborate character- and graphical-display devices are in open communication with the processor. A user can input his program almost instantaneously and, by executing or modifying it at will, he is able to interact in an experimental manner with the computational process.

4) Use of time-shared classroom display facilities to integrate the presentation of the theoretical and computational aspects of the course material.

5) Use of special laboratories having dedicated computers (i.e., reserved solely for this use) for part or all of the meetings of the class in order to integrate computational work directly into the instructional process.

6) Use of special laboratories for computer-aided instruction.

At present, the most frequently used approaches are those under 1), 2), and 3); for this program, 1) by itself is not satisfactory. Accordingly, we shall focus our discussion primarily upon the use of the batch mode 2) or the time-sharing mode 3).

No matter which type of computational service is chosen, the most essential points appear to be that it must be reliable, responsive to fluctuating student demands during a semester, and capable of allowing the student to complete assignments in a reasonable time span. In line with this, a complete dependence on slack-hour use of a computer owned by local industry, the shared use of campus equipment dedicated primarily to accounting and administrative functions, or the "generous" gift of an outdated computer will generally prove unsatisfactory.

For most of the requirements of this program, computational services in the batch mode can be entirely satisfactory, effective, and at the same time economical. One of the critical factors is then the "turnaround time" between the submission of input and the return of the output to the originator. Since the completion of a problem by a student may require four to eight, or even more, machine runs, a turnaround time that allows at least two runs during a normal day appears to be rather desirable. (With the aid of multi-processing

systems, it is possible to achieve a turnaround time of a few minutes or less for short student runs.) Besides the turnaround time, another controlling parameter in batch service is the availability of ancillary equipment for producing and handling punched cards. Here queues easily develop which are not readily reduced without considerable cost. It may be hoped that this latter problem will be alleviated considerably by the development of less expensive mark-sensing or character-reading devices.

Time-sharing services have much to recommend them. However, their costs are generally higher than those of acceptable batch services. Moreover, they can also lead to considerable queueing problems if not enough consoles are available to the students. The critical parameter is the maximal number of terminals which can be sustained by the particular computer system without a significant degradation of the response time.

The repertoire of available computer languages is an important consideration for any computational service. For many of the requirements of this program, one scientific language such as FORTRAN, BASIC, ALGOL, PL/1, or APL is sufficient. In general, however, it is desirable that the student gain experience with more than one language, and in certain courses, such as C3 or CM3, additional languages such as SNOBOL are particularly important. In several courses, including, for instance, CM1, CM2, or CM4, plotting and display facilities could also play a useful role. Indeed, here a versatile time-shared classroom display system of the type mentioned under 4) might be ideal and could completely determine the character of the courses. However, more modest services can be completely successful.

Broadly speaking, the computational services required by this kind of program can be provided in one or a combination of the following ways:

1) Use of off-campus computing facilities

2) Participation in an educational computer network

3) Operation of a campuswide educational computer facility

4) Operation of separate computer laboratories by different departments

Except under special circumstances, exclusive dependence on the first of these approaches is, in the long run, not very satisfactory. However, certain supplementary off-campus computer services, if reliable and economical, can provide highly advantageous solutions to enriching more modest services available on the campus

At present, educational computer networks have been established in only a few geographical locations. The organization of these networks ranges from fairly loose mutual assistance groups to highly organized hardware networks. Either time-sharing or batch-processing

services can be provided--sometimes both. Clearly, the access to a large central computer with a massive program library, large memory, fast central processor, and large systems and programming staffs represents a considerable advantage. On the other hand, logistical and communication problems, lack of control, etc., may turn out to be very detrimental for a participant college. Nevertheless, the possibility of joining such a network when feasible certainly deserves proper consideration.

Probably the most common approach toward meeting the educational computer needs of a college or university is the establishment of a centralized, campuswide academic computing center. Such a center will serve its expected purpose only when operated by an adequate staff in an efficient, professional manner; this is a point too often overlooked.

In recent years many small and medium-sized computers have been marketed at relatively low prices. This has made it possible for many institutions to have separate computers of varying sizes for individual departmental use. The assured availability of a specialized service to the department is, of course, one of the greatest advantages of this approach. It also allows the development of special laboratories of the type mentioned under 5) above. On the other hand, the computational work possible on these machines is severely limited by their size, and for more sophisticated tasks additional computer services are often needed.

The actual costs of a computing facility depend upon many factors, including the desired quality of the service, the intended group of users, the specific type of equipment selected, local physical facilities, and the corresponding staff needs. The Panel therefore decided not to include here any cost estimates for the facilities needed in this program. Some data on such costs are given, for example, in recent reports of the Southern Regional Education Board and the American Council on Education.*

* See Guidelines for Planning Computer Centers in Universities and Colleges and Computers in Higher Education, both publications of the Southern Regional Education Board, 130 Sixth Street, N.W., Atlanta, Georgia 30313. See also Computers on Campus, American Council on Education, One Dupont Circle, Washington, D. C. 20036, and "A Survey of Computing Costs," CRICISAM Newsletter 3, September, 1971, pp. 2-5.

4. Detailed Course Outlines

In this section we present outlines for the seven courses CM1, CM2, CM3, CM4 and C1, C2, C3. They are intended to suggest topics which might be included in these courses and should not be interpreted as check lists of required material. Where appropriate, suggested numbers of lectures to be spent on the various topics are included in the descriptions. These lectures total approximately 36 hours for a semester course; this leaves room for examinations, reviews, and lectures on supplementary technical material.

CM1. Computational Models and Problem Solving

Prerequisite: C1

Depth of treatment for the topics outlined below will vary with the interest of instructor and students, and lecture hours are therefore not assigned to any of the topics. It is recommended that a small number of fairly substantial projects be required in this course, rather than a larger number of smaller problems. Some of the material is suitable for group projects.

Detailed Outline

Statistical calculations

- Tabulation of data
- Calculation of means and variances
- Least squares fitting of straight lines
- Intuitive meaning of randomness
- Random number generators
- Tests of generators (e.g., chi-square)

Simulation of random processes

- Queues, inventories, random walks, etc.
- Discussion of statistical significance (confidence intervals)
- Games such as blackjack and bingo
- Monte Carlo calculations

Simulation of nonrandom processes

- Simple hypothetical computer
- Approximations to physical, economic, and biological processes
- Discussion of errors in such approximations
- Deterministic games such as nim

Other nonnumerical problems

- Enumeration
- Searching and sorting

Connectivity of graphs, shortest paths
Text editing
Elementary computer graphics
Handling arithmetic expressions

Sample Problems

1. Develop a program for the least squares fitting of straight lines to given data. The program should input pairs of values (x_i,y_i) and output the values of (a,b) where $y = ax + b$ is the best fit. Is the same line obtained when the values of x and y are interchanged? Show how your program can be used to fit curves given by $y = ab^x$ or $y = ax^b$ by taking the logarithms of each side of these equations. Are the results the same as those obtained by a true least squares fit of these curves without taking logarithms?

2. Write a program to generate 1000 pseudo-random numbers and calculate the chi-square statistic that is associated with 10 equal subintervals of the interval in which the random numbers are supposed to be uniformly distributed. If the numbers are random, the value of this statistic should exceed 16.9 with a probability of only 5 per cent. On the basis of this test, have you any reason for doubting the usefulness of your generator?

3. One relatively simple game of solitaire begins with a deal of nine cards, face up. If any two of these cards have the same face value, they are covered with two new cards, also face up. The last step is repeated until the deck has been exhausted except for one card, in which case the dealer has won the game, or until there are no more pairs showing, in which case the dealer has lost. Write a program to simulate this game and use it to determine an approximation to the probability of winning. How reliable do you believe the approximation to be?

4. Describe a model of cars moving through a highway toll station, and write a program to simulate the process. Use it to find approximations to the average delay and show how this delay depends on traffic density. Discuss the main limitations of your model. Assuming that one has a good model, what further limitations are there in the results obtained from any such simulation?

5. Write a program to simulate a game of blackjack and use it to compare different strategies. (This problem can be used as the basis for a group project.)

6. Describe a simple hypothetical computer and write a program to simulate its behavior. The description of the machine should be carefully documented so that any potential user will be able to determine exactly what the machine will do in every conceivable circumstance.

7. A man starts at the southwest corner of a field and runs north at 15 feet per second. His dog starts at the southeast corner,

200 feet from where the man starts, and runs directly towards his master at the rate of 40 feet per second. Calculate an approximation to the dog's path and to the time taken by the dog to catch his master. Compare this time with the time required if the shortest path had been taken.

8. Suppose that the adjacency matrix for a graph is given, along with two of its nodes. Write a program that will determine whether or not there is a path between the two nodes. Develop a second program for the same task, but based on a distinctly different algorithm, and compare the relative merits of the two different programs.

9. Develop a program for right-justifying text material. Input to the program should be a paragraph of text, and the corresponding output should be the same paragraph properly justified. (This problem can be expanded into a more substantial project on text editing by including additional features such as section headings and paging.)

10. A package of programs is to be developed for producing sequences of pictures. The pictures are to be output on a printer and must therefore be relatively simple, but the basic ideas are similar to those needed for computer-produced movies. (This can be a good group project. Once agreement is reached on how to represent the data, members of the group can be assigned separate tasks, such as developing subprograms for input, output, moving, shrinking, and rotating pictures.)

Bibliography

Most introductory books on computer programming contain material on computer applications. Some of these texts are cited in the outline of course C1 below. The following texts are primarily concerned with computer application problems suitable for this course:

Barrodale, Ian; Ehle, Byron L.; Roberts, F. D. K. Elementary Computer Applications in Science, Engineering, and Business. New York, John Wiley and Sons, Inc., 1971.

Gruenberger, Fred and Jaffray, George. Problems for Computer Solution. New York, John Wiley and Sons, Inc., 1965.

Hull, Thomas E. and Day, David D. F. Computers and Problem Solving. Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1970. (in particular, Part 2)

CM2. Introduction to Numerical Computation

Prerequisites: C1, M2, M3

Each of the major topics in the course should be amply motivated by introducing applications from the physical and social sciences. A consideration of electrical networks or input-output systems in economics leads, for instance, to linear systems; vibration problems from mechanics or Markov processes provide examples for eigenvalue problems; root-locus problems arise in several areas of engineering; observational data collected in practical experiments lead to a consideration of interpolation and least squares techniques. A selection of problems can be found, for example, in the book by Carnahan, Luther, and Wilkes (see bibliography).

During the course the students should solve a number of problems on the computer. Some of these should involve programming of the simpler algorithms and others should make use of library subroutines.

Detailed Outline

Introduction (2 lectures)

- Number representation on a computer
- Computer arithmetic
- Discussion of the various types of errors

Linear systems of equations (9 lectures)

- Gaussian elimination and the LU factorization
- Partial and complete pivoting
- Example of ill-conditioning
- Discussion of ways for detecting ill-conditioning
- The Wilkinson backward error result and its implications (no proofs)
- Iterative improvement
- Iterative methods with simple convergence criteria (no proofs)

Solution of a single nonlinear equation (6 lectures)

- Successive approximation
- The Point of Attraction Theorem and its implications
- Discussion of the rate of convergence
- Newton's method and the simplified Newton method
- Secant method and method of false position
- Stopping criteria for iterations
- Extension of Newton's method to two equations in two unknowns
- Roots of polynomials
- Sturm sequences
- Example of ill-conditioning of the roots of a polynomial

Interpolation and approximation (6 lectures)

Lagrange interpolating polynomial
Error term for an interpolating polynomial
Newton forward and backward difference polynomials
Piecewise polynomial interpolation
Least squares approximation, including numerical problems associated with the normal equations and orthogonal polynomials and their use in least squares
Chebychev economization of power series

Numerical differentiation and integration (6 lectures)

Error in differentiating the interpolating polynomial
Differentiation by extrapolation to the limit
Integration formulas based on interpolating polynomials and the associated error terms
Romberg integration
Gaussian quadrature formulas
Adaptive methods

The eigenvalue problem (6 lectures)

Direct root-finding methods such as Muller's or the secant method
The power method for the dominant eigenvalue
Subdominant eigenvalues by the inverse iteration method
The Householder-Givens method for symmetric matrices (without proofs)

Bibliography

Carnahan, Brice; Luther, H. A.; Wilkes, James O. Applied Numerical Methods. New York, John Wiley and Sons, Inc., 1969.
Primarily as a source of problems.

Conte, Samuel D. Elementary Numerical Analysis: An Algorithmic Approach. New York, McGraw-Hill Book Company, 1965.

Fox, Leslie and Mayers, D. F. Computing Methods for Scientists and Engineers. New York, Oxford University Press, Inc., 1968.

Fröberg, Carl E. Introduction to Numerical Analysis, 2nd ed. Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1969.

Henrici, Peter K. Elements of Numerical Analysis. New York, John Wiley and Sons, Inc., 1964.

McCracken, Daniel D. and Dorn, William S. Numerical Methods and FORTRAN Programming. New York, John Wiley and Sons, Inc., 1964.

Stiefel, E. L. An Introduction to Numerical Mathematics. New York, Academic Press, Inc., 1963.

Wendroff, Burton. First Principles of Numerical Analysis. Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1969.

CM3. Combinatorial Computing

Prerequisites: C1 and M3

The material listed here may be more than can be covered properly in one semester. Since many topics are rather independent of each other, an instructor can make his own selection of what to exclude. For this reason no breakdown into the number of lectures for each topic was included. Students are expected to implement some of the algorithms on the computer and also to experiment with relevant library subroutines. For this computational work, it may be desirable to assign team projects rather than to let every student proceed on his own.

Detailed Outline

The machine tools of combinatorics

- Integers and their representation, including radix, modulo, and factorial representation (and its use in indexing over permutations), monotonic vector representation (and its use in indexing over combinations and partitions)
- Sets and their representation, including bitstring and index representation
- Some aspects of list processing and storage organization, including representation of variable length sequences, one- and two-way lists, tree structures, free storage, and garbage collection

Enumeration and counting

- Enumeration techniques, such as backtrack and sieve methods
- Counting techniques, including recurrence relations and techniques for solving them, Pólya's counting formula

Sorting

- Internal sorting; insertion, selection, and enumeration methods
- External sorting; long-sorted subsequences, merging, distribution sorting

Searching

- Searching in a linearly ordered set, including hash-coding or scatter storage techniques, Fibonacci search
- Trees and their use in ordering sets, rooted trees and their properties, representation of trees, methods of traversing trees, internal and external path length, optimal and near optimal search trees

Heuristic search, game trees, minimax evaluation, pruning, static evaluation functions, backing up uncertain values

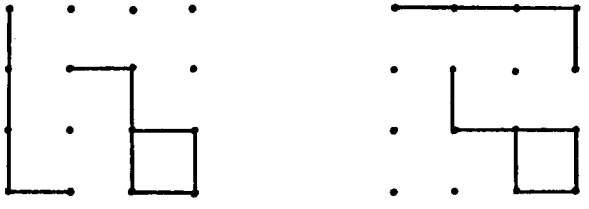
Graph algorithms

Some concepts from graph theory, such as graphs, directed graphs and their representation, paths, trees, circuits and cutsets
Connectedness and shortest path problems, including various related algorithms
Flow problems, max-flow and min-cut theorem, Ford-Fulkerson algorithm
Spanning trees, and algorithms for finding them
Graph isomorphisms
Planarity of graphs

Sample Problems

1. In how many different ways can one color the six faces of a cube which may be freely rotated with two colors? [Topics: counting, group of transformations, Pólya's theorem]

2. An integrated circuit manufacturer builds chips with 16 elements arranged in a 4 x 4 array as shown below. To realize different circuits all patterns for interconnecting the elements are needed. Direct interconnections are made only between horizontally or vertically adjacent elements, e.g., as shown below:



(Closed loops do not usually occur, but this is ignored here for simplicity's sake.) To deposit interconnections on the chip a photo-mask of the interconnection pattern is needed. Notice that the same photo-mask will do for the two interconnection patterns shown above. How many photo-masks are required in order to lay out all possible interconnection patterns on these chips?

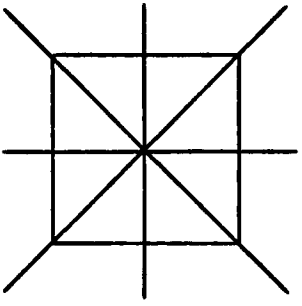
a) Carefully define the permutation group involved.

b) Solve the problem using Burnside's lemma alone.

c) Solve the problem using Pólya's counting formula.

[Topics: counting, group of transformations, Pólya's theorem]

3. List all the essentially different ways in which eight queens can be placed on a chessboard so that no two are on the same row, column, or diagonal. Two ways of placing queens are essentially different if they cannot be transformed into each other by a rotation of the board or by reflection on any of the axes shown in the figure:



4. Assume a large deck of N punched cards is dropped on the floor, but fortunately each card contains a unique sequence number from 1 to N which indicates its position in the deck. After the cards have been picked up, the deck is not in complete disorder; it contains long runs of cards in proper order. Discuss what sorting techniques can be considered to sort the deck as efficiently as possible. What standard sorting techniques would definitely be inefficient in this case? [Topics: linear order, expected number of comparisons, sorting algorithms]

5. a. Prove that every positive integer A has a unique representation $a_1, a_2, \ldots, a_n$ which satisfies the conditions

(i) $A = a_1 \cdot 1! + a_2 \cdot 2! + \ldots + a_n \cdot n!$;

(ii) $0 \leqq a_i \leqq i$ for $i = 1, 2, \ldots, n$;

(iii) $a_n \neq 0$.

Let the factorial representation for zero be $a_1 = 0$, so that $0 = 0 \cdot 1!$.

b. Devise an algorithm for adding 1 to a number in factorial representation.

c. Devise algorithms for adding and subtracting two numbers in factorial representation.

d. For fixed $N \geq 1$, there are $(N+1)!$ numbers whose factorial representation $a_1, a_2, \ldots, a_n$ has $n \leq N$. From this fact and from the uniqueness of the factorial number representation proved in (a), derive the identity:

$$1 \cdot 1! + 2 \cdot 2! + 3 \cdot 3! + \ldots + N \cdot N! = (N+1)! - 1$$

e. The factorial number representation is useful in enumerating permutations. This can be done in many ways. The technique discussed below is called the Derangement Method of M. Hall.

Let $P = (i_0, i_1, \ldots, i_N)$ be a permutation of the $N + 1$ integers $0, 1, \ldots, N$. For $j = 1, 2, \ldots, N$ define:

a_j = (the number of integers $< j$ which occur to the right of j in permutation P)

As an example, the permutation $P = (2, 0, 1)$ yields $a_1 = 0$, $a_2 = 2$.

By considering $a_1, \ldots, a_N$ to be the factorial representation of an integer A, we have set up a correspondence between the $(N+1)!$ permutations of the integers $0, 1, \ldots, N$ and the $(N+1)!$ numbers with factorial representation $a_1, \ldots, a_n$ $(n \leq N)$.

(e_1) Prove that this correspondence is 1:1.

(e_2) Devise an algorithm which constructs the permutation associated with an integer A from the factorial representation of A.

6. Devise an algorithm for finding shortest paths in a graph with weighted nodes. The length of a path is defined to be the sum of the weights of all nodes which lie on the path.

Consider the following three variations of the problem:

a. paths between two given nodes

b. paths between one given node and all other nodes

c. paths between all pairs of nodes

[Topics: shortest paths, wave propagation algorithm]

Bibliography

There are several good books on combinatorial mathematics in general and on graph theory in particular, but there appears to be none which is written from the point of view proposed here, of emphasizing the computational aspects of algorithms for solving combinatorial problems.

The book that comes closest to this point of view is

Beckenbach, Edwin F., ed. Applied Combinatorial Mathematics. New York, John Wiley and Sons, Inc., 1964.

Much useful material on computational and programming aspects of algorithms, combinatorial ones in particular, can be found in:

Knuth, Donald E. The Art of Computer Programming. Reading, Massachusetts, Addison-Wesley Publishing Company, Inc.
Vol. 1. Fundamental Algorithms, 1968.
Vol. 2. Seminumerical Algorithms, 1969.
Vol. 3. Sorting and Searching, 1971-72.

The following references are not intended to be exhaustive by any means, but simply to point to a few papers which are typical of those which concentrate on computational aspects of combinatorics.

The machine tools of combinatorics

Hall, Marshall, Jr. and Knuth, Donald E. "Combinatorial analysis and computers." American Mathematical Monthly, 72 (1965), pp. 21-28.

Lehmer, Derrick H. "The machine tools of combinatorics." In Beckenbach, Edwin F., ed. Applied Combinatorial Mathematics. New York, John Wiley and Sons, Inc., 1964.

Lehmer, Derrick H. "Teaching combinatoric tricks to a computer." Proceedings of Symposia in Applied Mathematics, 10. Combinatorial Analysis, pp. 179-194. Providence, Rhode Island, American Mathematical Society, 1960.

Enumeration and counting

Golomb, Solomon W. and Baumert, Leonard D. "Backtrack programming." Journal of the Association for Computing Machinery, 12 (1965), pp. 516-524.

Lehmer, Derrick H. "The sieve problem for all-purpose computers." Mathematical Tables and Other Aids to Computation, 7 (1953), pp. 6-14.

Swift, J. D. "Isomorph rejection in exhaustive search techniques." Proceedings of Symposia in Applied Mathematics, 10. Combinatorial Analysis, pp. 195-200. Providence, Rhode Island, American Mathematical Society, 1960.

Walker, R. J. "An enumerative technique for a class of combinatorial problems." Proceedings of Symposia in Applied Mathematics, 10. Combinatorial Analysis, pp. 91-94. Providence, Rhode Island, American Mathematical Society, 1960.

Searching

Hibbard, Thomas N. "Some combinatorial properties of certain trees with applications to searching and sorting." Journal of the Association for Computing Machinery, 9 (1962), pp. 13-28.

Morris, Robert. "Scatter storage techniques." Communications of the Association for Computing Machinery, 11 (1968), pp. 38-44.

Peterson, W. W. "Addressing for random access storage." IBM Journal of Research and Development, 1 (1957), pp. 130-146.

Graph algorithms

Corneil, D. G. and Gottlieb, C. C. "An efficient algorithm for graph isomorphism." Journal of the Association for Computing Machinery, 17 (1970), pp. 51-64.

Dijkstra, E. W. "A note on two problems in connexion with graphs." Numerische Mathematik, 1 (1959), pp. 269-271.

Edmonds, Jack. "Paths, trees and flowers." Canadian Journal of Mathematics, 17 (1965), pp. 449-467.

Gottlieb, C. C. and Corneil, D. G. "Algorithms for finding a fundamental set of cycles for an undirected linear graph." Communications of the Association for Computing Machinery, 10 (1967), pp. 780-783.

Lee, C. Y. "An algorithm for path connections and its applications." Institute of Radio Engineers Transactions on Electronic Computers, EC-10 (1961), pp. 346-365.

Moore, Edward F. "The shortest path through a maze." Proceedings of the International Symposium on the Theory of Switching, pp. 285-292. Cambridge, Massachusetts, Harvard University Press, 1959.

Warshall, Stephen. "A theorem on Boolean matrices." Journal of the Association for Computing Machinery, 9 (1962), pp. 11-12.

CM4. Differential Equations and Numerical Methods

Prerequisites: CM2, M4

Throughout this course it is desirable to introduce problems which lead to the types of equations considered at the time. Excellent sources include circuit theory, mechanical systems, biological systems, particle dynamics, and economics. Numerical methods are to be introduced early in the course both to illustrate the qualitative

behavior of solutions and to motivate uniqueness and existence arguments. In considering these methods the student should be made aware of the effects of discretization--and roundoff errors--and of stability. The students are expected to write some programs for various methods and to use existing library subroutines for others.

Detailed Outline

Origin and examples of differential equations (2 lectures)

- Sample (deterministic and nondeterministic) problems from the physical, social, and biological sciences, including predator-prey model
- Difference equations, including examples of different equations leading to the same differential equation

Simple linear equations (4 lectures)

- $y' = f(x)$, $y' = ay + f$, $y'' = ay' + by + f$
- Representation of solutions by indefinite integrals and special functions
- Direction fields
- Qualitative behavior of solutions
- Uniqueness and continuous dependence on initial data
- Consequences of linearity
- Approximation by Taylor series
- Polygon method
- Trapezoidal approximation
- Equivalence of second-order equations to first-order systems
- Introduction to first- and second-order difference equations and their elementary properties

The first-order equation $y' = f(x,y)$ (9 lectures)

- Graphical treatment, polygon method
- Relation to integral equations, Picard iteration
- Quadrature methods
- Picard existence and uniqueness theorem with proof
- Statement of Peano existence theorem
- Nonuniqueness examples
- Discussion of continuous dependence on initial data
- Power series solution and numerical methods
- Runge-Kutta methods
- Predictor-corrector methods
- Discussion on consistency and convergence (without proofs)

First-order systems of equations (8 lectures)

- Redevelopment for first-order systems--using vector notation--of the major results about single first-order equations
- Review of matrix results, similarity transformations, series for $\exp(At)$ and semigroup properties
- Vector space of solutions of $y' = Ay$, the adjoint solution

Representation of solutions of nonhomogeneous problems
Stiff systems

Plane autonomous systems (7 lectures)

Numerical exploration of $y' = ax + by + f(x,y)$, $x' = cx + dy + g(x,y)$
Poincaré phase plane and critical solutions
Critical points and concepts of stability
Numerical comparison of linear and nonlinear equations
The Lienard equations
Liapounov's ideas
Exploration of predator-prey model

Two-point boundary value problems (6 lectures)

Exploration of the linear second-order equation with mixed boundary conditions by shooting techniques
Discretization and methods for solving the resulting equations
Extensions to nonlinear equations

Bibliography

Birkhoff, Garrett and Rota, Gian-Carlo. Ordinary Differential Equations. Boston, Massachusetts, Ginn and Company, 1962.
Selected topics.

Daniel, James W. and Moore, Ramon E. Computation and Theory in Ordinary Differential Equations. San Francisco, California, W. H. Freeman and Company, 1970.

Henrici, Peter. Discrete Variable Methods in Ordinary Differential Equations. New York, John Wiley and Sons, Inc., 1962.

Keller, Herbert B. Numerical Methods for Two-Point Boundary Value Problems. Boston, Massachusetts, Ginn and Company, 1968.
Advanced discussion of material on two-point boundary value problems.

Lapidus, Leon and Seinfeld, John H. Numerical Solution of Ordinary Differential Equations. New York, Academic Press, Inc., 1971.

C1. Introduction to Computing

Prerequisite: College admission

As stated in Section 2, this course should be oriented toward problem solving with computers. Accordingly, it is important that, throughout the course, different types of problems are considered and appropriate algorithms for their computational solution are designed and discussed. In particular, it is essential that both numerical and nonnumerical applications are presented. The problems

should be reasonably interesting and realistic, and some should be open-ended, requiring a certain effort to identify what is required and how the solution is to be obtained. At least one major project leading to a completely verified and documented program should be included.

The course can serve to introduce many traditional mathematical ideas from a different point of view (e.g., subroutines and functions, induction and recursion, etc.). Such identifications should be strengthened where possible.

The course should be organized so that students can write small computer programs almost immediately. This may be accomplished by representing algorithmic processes from the outset both by flowcharts and programming languages.

The following outline is for a one-semester course meeting three times each week for lectures. In addition, it is generally advisable to schedule a regular weekly laboratory period of at least two hours. No lecture hours were assigned since the need for proper sequencing of programming assignments often demands that certain topics are either interchanged or distributed throughout the course.

Detailed Outline

Problems, algorithms, and programs

- Typical problems and mathematical models
- Concept of an algorithmic process
- Flowcharts
- Basic structure and properties of algorithms
- Concept of a program
- How computers execute programs
- Elements of a higher-level programming language

Basic programming

- Number and character representation
- Constants and variables
- Principal syntactic statements of the language
- Functions, subroutines, and complete programs
- Elements of the system being used
- Libraries
- Program testing and documentation

Errors and approximations

- The approximate character of mathematical models
- Truncation and roundoff error
- Verification of algorithms
- Error conditions and messages
- Techniques for algorithm testing
- The idea of numerical stability

Data structures

Discussion of a variety of problems leading to different data structures such as vectors, arrays, strings, trees, linked structures
Basic manipulation of the different structures

Advanced topics
Further details of the programming language
Aspects of compilers
Basic structure of an operating system
Aspects and organization of computer systems

Survey of computers, languages, and systems

Historical developments, discussion of different language types, aspects of systems programs, new developments

Bibliography

Arden, Bruce W. An Introduction to Digital Computing. Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1963.
A good reference for the instructor.

Cole, R. W. Introduction to Computing. New York, McGraw-Hill Book Company, 1969.

Forsythe, Alexandra I.; Kennan, Thomas A.; Organick, Elliott I.; Stenberg, Warren. Computer Science: A First Course. New York, John Wiley and Sons, Inc., 1969.
This is a text for a high school course but may be appropriate for this course.

Galler, Bernard A. The Language of Computers. New York, McGraw-Hill Book Company, 1962.
A good reference for the instructor.

Gruenberger, Fred. Computing: An Introduction. New York, Harcourt Brace Jovanovitch, Inc., 1969.

Hull, Thomas E. Introduction to Computing. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1966.

Hull, Thomas E. and Day, David D. F. Computers and Problem Solving. Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1970.
Part I of this text emphasizes material appropriate for this course.

Kemeny, John G. and Kurtz, Thomas E. Basic Programming. New York, John Wiley and Sons, Inc., 1967.
An introduction to programming with applications.

Rice, J. K. and Rice, J. R. Introduction to Computer Science: Problems, Algorithms, Languages, Information and Computers. New York, Holt, Rinehart and Winston, Inc., 1969.

Walker, Terry M. and Cotterman, William W. An Introduction to Computer Science and Algorithmic Processes. Boston, Massachusetts, Allyn and Bacon, Inc., 1970.

C2. Computer Organization and Programming

Prerequisite: C1

This course includes computational projects in assembly language programming. However, in line with the survey character of the course, care should be taken not to involve the students in a too-detailed discussion of assembly languages or of computer hardware. A scheduled laboratory period is desirable.

Detailed Outline

Computer structure and machine language (2 lectures)

- Fundamentals of computer organization, including registers, arithmetic units, memory, I/O units, and their interdependence
- Description of typical single-address machine instructions
- Programs as sequences of machine instructions and their execution

Introduction to symbolic coding and assembly systems (5 lectures)

- Mnemonic operation codes
- Labels, symbolic address
- Literals
- Pseudo operations
- General construction of assemblers
- Simple examples and exercises using a locally available assembler

Digital representation of data (3 lectures)

- Bits, fields, words
- Character representation
- Radix representation of numbers, radix conversion, representation of integers, floating point, and multiple precision numbers in binary and decimal form
- Variable length data

Addressing (2 lectures)

- Absolute addressing, indexing, indirect addressing, relative addressing
- Zero-, one-, two-, three-address instruction formats
- Address transformations

Machine organization to implement addressing structures
Character- versus word-oriented machines

Logic design (5 lectures)

Elements of Boolean algebra
AND, OR, NOT logic gates
Implementation of Boolean functions
Encoders and decoders
Descriptive discussion of clocked circuits, flip-flops, registers, shift registers, accumulators, counters, timing chains

Arithmetic units (3 lectures)

Serial versus parallel arithmetic
Implications of choice of radix
Design of a simple arithmetic unit
Design of half-adder and adder
Algorithms for multiplication and division

Instruction units (3 lectures)

Instruction fetch and decoding
Program sequencing
Branching
Subroutine calls
Interrupts
Control and timing logic
Micro-programming as a means of implementing control units

Storage units (3 lectures)

Structure of core memory
Typical memory bus structure
Memory overlap, protection, relocation, and paging
Word versus character organizations
Types of bulk memories
Descriptive discussion of stack memories, associative memories, read-only memories, and virtual memory schemes

Input-output systems (3 lectures)

Direct memory access I/O
I/O channels and controllers, multiplexers
Characteristics of various types of input/output devices
Relation of I/O system to control unit and main memory
Input/output programming
Buffering and blocking
Interrupts
Problems of error detection and correction in data transmission

Systems software (4-5 lectures)

Operating systems
Input/output packages
Assemblers, loaders
Interpreters, compilers
Utility programs and libraries

Survey of contemporary computers (3-6 lectures)

A survey of contemporary computers emphasizing a variety of machine organization. Typical topics: large versus small computers; single register, multiple register, and stack machines; unorthodox machines. Discussion of possible implementation of high-level programming language statements on typical computers.

Bibliography

Bell, C. G. and Newell, A. Computer Structures. New York, McGraw-Hill Book Company, 1970.
Survey of computer organizations. Source of material for the survey of contemporary computers.

Chu, Yaohan. Digital Computer Design Fundamentals. New York, McGraw-Hill Book Company, 1962.
A somewhat dated reference on logic design.

Gear, C. William. Computer Organization and Programming. New York, McGraw-Hill Book Company, 1969.
Reference on assembly language programming.

Gschwind, H. W. Design of Digital Computers: An Introduction, 5th ed. New York, Springer-Verlag New York, Inc., 1970.
Text on computer design and organization, slightly engineering-oriented.

Hellerman, H. W. Digital Computer System Principles. New York, McGraw-Hill Book Company, 1967.
Uses Iverson notation, directed toward IBM equipment, especially S/360.

Knuth, Donald E. The Art of Computer Programming. Volume 2, Seminumerical Algorithms. Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1969.
Reference for a mathematical treatment of computer arithmetic (Chapter 4).

McCluskey, E. J. Introduction to the Theory of Switching Circuits. New York, McGraw-Hill Book Company, 1965.
Reference for basic switching theory.

Nashelsky, Louis. Digital Computer Theory. New York, John Wiley and Sons, Inc., 1966.

A paperback containing a survey of many of the topics covered in this course.

C3. Programming Languages and Data Structures

Prerequisite: CM1

Detailed Outline

Structure of algorithmic languages (8 lectures)

- Review of basic program constituents of the language introduced in C1
- Introduction to the elements of ALGOL or PL/1
- Informal syntax and semantics of simple statements in that language
- Backus normal form
- Grouping of statements and block structure of programs
- Scopes, local and nonlocal quantities
- Functions and procedures
- Formal and actual parameters
- Binding time of program constituents
- Simple recursive procedures
- Concept of a stack
- Simulation of recursions as iterations using stacks

Arithmetic statements (4 lectures)

- Brief discussion of graphs and trees
- Tree diagrams of arithmetic expressions
- Informal discussion of precedence hierarchies
- Infix, prefix, postfix notation
- Translation between infix and postfix notation
- Evaluation of expressions in postfix notation

Trees and lists in a computer (8 lectures)

- Types of data nodes and linkages
- List names, list heads, sublists
- Multilinked lists
- Stacks as list structures with usage discipline
- Representation of trees as special cases of lists
- Accessing, insertion, deletion, and updating in trees
- Traversal schemes for trees
- Application to the generation of machine code from expression trees

String manipulation (7 lectures)

Introduction to a string manipulation language such as SNOBOL

Data declarations in such a language
Recursive algorithms in such languages
Applications to formal differentiation of expressions

Data structures and storage allocation (3 lectures)

Storage allocation for algorithmic language structures such as independent, nested blocks, strings, arrays, etc.
Procedures using run-time stacks
Storage allocation for string manipulation languages

Some aspects of languages and grammars (6 lectures)

Syntax, semantics, and pragmatics of programming languages
The concept of a formal grammar
Production notation
Discussion of Chomsky's classification of grammars
Discussion of computability, undecidability
Syntax and semantics of arithmetic statements
Precedence and operator precedence grammars
Syntactic specification of procedures, blocks, and statements
Formal semantics corresponding to syntactic specifications

Bibliography

Genuys, F., ed. Programming Languages. New York, Academic Press, Inc., 1968.

Harrison, M. C. Data Structures and Programming. Courant Institute of Mathematical Sciences, New York University, 1970.

Knuth, Donald E. The Art of Computer Programming. Volume I, Fundamental Algorithms. Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1968.
Important presentation of data structures.

Rosen, Saul, ed. Programming Systems and Languages. New York, McGraw-Hill Book Company, 1967.
Contains, among other things, a discussion of SNOBOL and a comparison of list processing languages.

Sammet, Jean E. Programming Languages: History and Fundamentals. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1969.
A comprehensive survey of languages.

Wegner, Peter. Programming Languages, Information Structures, and Machine Organization. New York, McGraw-Hill Book Company, 1968.
An approach to programming languages as information structures.

RECOMMENDATIONS ON UNDERGRADUATE MATHEMATICS COURSES INVOLVING COMPUTING

A Report of

The Panel on the Impact of

Computing on Mathematics Courses

October 1972

TABLE OF CONTENTS

1. Preface

The growing influence of modern electronic computing in many fields of knowledge has contributed to a dramatic increase and diversification in the application of mathematics to other disciplines. No longer are the uses of mathematics confined exclusively to the physical sciences and engineering; they are found with increasing frequency in the social, behavioral, and life sciences as well. Correspondingly, the use of the computer has led to different requirements for the solution process in mathematics itself. Theory construction and model building have assumed a different dimension; in addition to knowing existence theorems, the user of mathematics must know constructive methods for solving problems, and he must have the means to ascertain the efficiency as well as the correctness of these methods.

These developments have created new challenges with regard to revision of the undergraduate mathematics curriculum. The basic curriculum should reflect the contemporary points of view associated with computer application in mathematics; it should acquaint the students with the newly developed methods of solving standard problems and also introduce them to the host of problems which have arisen in the past few years.

There now appears to be growing recognition that more consideration should be given to the potential impact of the computer on the basic undergraduate courses which serve not only potential mathematicians and computer scientists but many other students as well. The present report is the result of the first study of this problem by CUPM.

Although a consensus about the role of computers in the basic mathematics curriculum has not yet evolved, we believe there is an urgent need for experimentation in this area. This report presents ideas for such experimentation by proposing changes in various basic mathematics courses and by suggesting some new courses which are designed to take advantage of the presence of computers.

As is the case with all CUPM reports, these recommendations must be regarded as general suggestions which will need to be adapted to local circumstances and revised in the light of subsequent experience. Nevertheless, mathematics departments should immediately concern themselves with the ideas outlined in this report so that they can prepare their students for the uses of mathematics in the context of the availability of computers.

2. Premises of this Study

We suggest four ways in which the computer can influence undergraduate mathematics education:

(i) Computing can be introduced into traditional mathematics courses;

(ii) New courses in computationally-oriented mathematical topics can be designed;

(iii) The entire curriculum can be modifed to integrate computing more fully into the student's program;

(iv) Computers and computer-related devices can be used as direct aids to mathematical instruction.

This report addresses itself to (i) and certain aspects of (ii). As for (iii), some possible curriculum restructuring relating to computing has been discussed in the CUPM report Recommendations for an Undergraduate Program in Computational Mathematics. Finally, (iv) is a very broad area which would require a separate study of its own; we include only a brief discussion of some related topics in the final section of the report.

We do not suggest that all mathematics instruction be modified along any of these lines. In recent years it has become appropriate to speak of the mathematical sciences in a broad sense rather than of mathematics in the more familiar, narrower sense. This situation indicates a need for different avenues within mathematics education; the introduction of computer-oriented material should therefore be regarded as a development parallel to the standard curriculum which interacts with the standard curriculum at a number of places.

Nearly every student taking mathematics courses can benefit from some computer-oriented mathematics instruction. The use of computers is beginning to pervade all phases of life in our society, and in most disciplines, including mathematics, there is a need for students to become familiar with some aspects of computing. Many mathematics departments have observed that well over half of their undergraduate majors enter computer-related careers or graduate programs after graduation. Clearly, these students would benefit considerably from computer-oriented courses and curricula. Computer-oriented courses also serve all those students from other disciplines who are interested in learning more mathematics in order to solve problems from their own fields. Modern applied mathematics has a strong computer orientation; when students enter this field, those whose education stresses concern for computational problems have a decided advantage over those who are familiar only with theoretical results. Finally, the growing trend toward introducing computers in high schools will require that prospective teachers learn about the interaction between the computer and mathematics.

Although the content and objectives of computational material differ considerably for various groups of students, computing provides all of them an unusual opportunity for active participation. For this reason the motivational aspects of computing are significant for most students, and the value of such motivation should not be underestimated.

A more substantive objective must be to select course material and approaches so as to reflect the actual influence of computing on mathematics. The recommendations which follow are based on the premise that any program which seeks to reflect this influence should stress four points--namely, algorithms, approximations, model building, and the nature of the entire problem-solving process.

Algorithms. The modern computer's development as a general-purpose problem-solving system derives not so much from its arithmetic capabilities but from its ability to handle logical and nonnumerical problems. From a mathematical viewpoint this has led to a greater emphasis on the construction and analysis of algorithms for the actual solution of mathematical problems rather than only on the proof of the existence of solutions. Stressing the algorithmic aspects forces the student to state both the problem and the method of solution in precise and unambiguous terms. It fosters his ability to organize and formulate logically an attack on a problem as well as to recognize and clarify the assumptions he is making in order to solve the problem.

Approximations. In most analysis courses numerical algorithms are more prevalent than nonnumerical ones. This leads to questions of error or, more generally, to questions about the quality of the approximations produced by the algorithm. If an algorithm produces an answer, some statement is needed to relate this answer to a solution of the original mathematical problem. If a process appears to converge, there is a need to prove that the process converges as well as to determine how rapidly it converges. If a method is bound to be applicable to certain input data, it is necessary to establish what happens when changes are introduced in these data. Clearly, in undergraduate courses these questions can rarely be answered satisfactorily but the student should acquire a concern for them and an appreciation of their importance.

Model Building. An important part of every real application of mathematics is the recognition and formulation of a satisfactory mathematical model of the given nonmathematical problem. Developing the student's skill in this process should be an objective of every course involving computer applications.

The Problem-Solving Process. Modeling, the development of algorithms, the study of the approximations used, and the computation and interpretation of results are all principal steps in the process of solving a problem on a computer. It is important to stimulate in the student an understanding of this process viewed as a whole by discussing and assigning the complete solution of appropriate simple problems.

The four points raised here--namely, the stress on algorithms, the development of a concern for the quality of approximations, the emphasis on model building, and a general emphasis on the entire problem-solving process--should be considered as general objectives in the student's program. In those undergraduate courses which involve a limited amount of computing, little more can be done than to illustrate the importance of these points and to instill in the student an intuitive understanding and concern for them. This requires a careful selection of the computational topics to be discussed and of the problems to be assigned. In other words, without underestimating the motivational value of computing, we believe that the introduction of computational material into a mathematics course should go beyond merely illustrating a mathematical concept. It should at least provide a definite answer to a specific question or problem in order to give the student deeper insight into the theory, the model, and the algorithms used.

Implementation and Precautions. There are wide variations in the extent and type of computer use which can be introduced into a traditional mathematics course. Such variations arise both from differences in computing facilities and from differences in the instructors' opinions of how essential these uses can and should be for the course. Where computing facilities are readily available or where courses are modified extensively to emphasize the four objectives discussed above, the trend is often to use the computer frequently and in a matter-of-fact manner. This means that computer-related material is presented throughout the course and that computer problems are assigned as a regular part of the homework; the student is expected to master these problems in order to have a coherent understanding of the subject. Where computing facilities are not so readily available or where computer-related material enters the course only in a secondary, supportive role, the trend is to consolidate computer use into the solution of a number of relatively substantial problems and to expect the student to apply his mathematical knowledge to these problems, but not to demand mastery of these problems for a coherent picture of the course. In this case, extra credit is sometimes given for the computer component of a course.

In whatever way the computer is used, there are a number of precautions which ought to be observed. Primarily, one should neither misuse nor overuse the computer. The computer is certainly misused when one is not mathematically honest about what it can or cannot do. For example, a computer can approximate a limit, but it cannot "compute" one or verify its existence, nor can it "test" a function for continuity. Specific examples of overuse of the computer are harder to provide, but it can be recognized when computing begins to crowd mathematical material out of the course or when students become bored by it. Overuse of the computer can result when the excitement of the new approach obscures the principal purpose: to teach mathematics. It is primarily the algorithmic approach together with the other three objectives, rather than the actual use of a computer, which will help to advance this purpose. Many points about algorithms can be made without using a computer at all; three-

digit arithmetic can be used to discuss approximations and roundoff error, and model-building and problem-solving expertise can also be gained from judiciously chosen paper-and-pencil problems. Also, students need not program every algorithm they encounter, and experiments with preprogrammed algorithms can often provide more insight than the lengthy drudgery of debugging a complicated program. It is up to the individual instructor to maintain a proper balance between the use of the computer and the other components of the course.

3. Basic Courses

3.1 Introduction

In this section we discuss five one-semester beginning undergraduate courses which include an emphasis on computing. As in all CUPM reports, the outlines given here are not meant to be prescriptive but are intended to extend the exposition of our ideas in the previous section by giving suggestions and possible approaches for implementing them.

For reference purposes we begin with a list of brief catalog descriptions for these courses. The descriptions do not include any programming requirements; these are discussed in Section 5.2.

MC-0. Elementary Functions and Problem Solving. [Prerequisite: College admission] Basic computer programming, elementary functions, matrix operations. These topics are to be motivated by, and applied to, practical problems.

MC-1. Calculus I with Computer Support. [Prerequisite: MC-0 plus trigonometry, or equivalent mathematical background] Differential and integral calculus of the elementary functions with associated analytic geometry, supported by computer applications.

MC-2. Calculus II with Computer Support. [Prerequisite: MC-1] Techniques of integration, introduction to multivariate calculus, and elements of differential equations, supported by computer applications.

MC-DM. Discrete Mathematics. [Prerequisite: No specific course prerequisite, but see page 590] Concepts and techniques in discrete mathematics that find frequent applications in computing problems.

MC-3. Algorithmic Elementary Linear Algebra. [Prerequisite: MC-0 or equivalent background] An introduction to matrix and vector algebra in n dimensions with an emphasis on algorithmic aspects.

The four courses MC-0 through MC-3 represent computer-oriented versions of the courses Mathematics 0 through 3 in the 1972 CUPM report Commentary on a General Curriculum in Mathematics for Colleges (GCMC Commentary). In the case of MC-0 and MC-3, the material in the GCMC Commentary courses was considerably modified and rearranged in order to introduce a fairly strong emphasis on computation. In MC-1 and MC-2, on the other hand, the purpose and the outline have remained essentially the same as for traditional courses; the emphasis on what is taught, however, has shifted along with the shift in the kinds of applications that are possible with the computer. The remaining course MC-DM represents a new development. It has a strong algorithmic flavor and introduces material of considerable importance in many computer applications. The course may not only supplement the standard curriculum but could also serve well as a first mathematics course for students from many disciplines.

Ideally, a student entering any of these computer-related courses other than MC-0 should have at least a rudimentary knowledge of programming. Since this is, at present, an unrealistic requirement, several possible alternatives are suggested in Section 5.2. Since these alternatives depend strongly on local circumstances, no further mention of them is made in the outlines. Only in the case of MC-0 is some time allotted to introduce certain elementary computer concepts.

In each of the following outlines, the suggested pace is indicated by assigning a number of hours to each group of topics. A standard semester contains 42 to 48 class meetings, and we follow the GCMC Commentary in allowing approximately 36 hours for discussion of new material; the remaining time can be devoted to tests, reviews, etc.

3.2 Course Outlines

MC-0. Elementary Functions and Problem Solving

[Prerequisite: College admission] The aim of this freshman-level course is to teach students ways to approach problems in the physical, natural, and social sciences and to equip them with some fundamental mathematical and computational tools for the solution of these problems. Typical problems are concerned with measurement and prediction: given a process such as a factory producing steel, traffic moving on a city street, or a shifting population, it is desired to predict future properties of the process on the basis of past measurements. The approach taken in the course is first to have students model and simulate specific processes using a computer and then look for functional relationships between various aspects of these processes, e.g., between time and the total output of steel,

between traffic light timing and traffic density, or between past and future population distributions. The objectives of this approach are to create an understanding of modeling through approximation and simulation and a feeling for the types of questions asked about models and functions.

Studies of elementary functions, computational techniques, and matrix operations are interwoven in the course and are used to illustrate and motivate one another. Depending upon the selection and treatment of particular topics, this course may serve either as a refresher course for students going on to calculus or as a terminal course for students who intend to take only one course in mathematics. However, this course alone probably will not prepare students adequately for a one-year sequence in calculus, since it does not contain the topics from trigonometry which they will need. The orientation towards applications and the emphasis on computing should make the course attractive to many students who might otherwise avoid more traditional mathematics courses. There are no prerequisites, as instruction in the use of a computer is integrated with the rest of the course.

COURSE OUTLINE

1. Introduction. (6 hours) Number representation, algorithms, elements of programming, functions, relations.

2. Linear and quadratic functions. (8 hours) Simulations involving constant and accelerated rates of change, graphs of linear and quadratic functions, zeros, maxima, minima, applications.

3. Linear programming. (4 hours) Linear functions of two variables, linear inequalities, maxima, minima, applications.

4. Matrix operations. (6 hours) Representations of tabular data, subscripts, matrix and vector operations, simultaneous equations, applications.

5. Algebra of functions. (6 hours) Algebraic operations on functions, polynomial and rational functions, maxima, minima, zeros, inverses, composition.

6. Exponential and logarithmic functions. (6 hours) Simulations of exponential growth, properties of exponents, logarithms as inverses of exponentials.

1. Introduction. Simple mathematical concepts can be introduced or reviewed in the context of teaching the rudiments of programming in a computer language such as APL, BASIC, or FORTRAN. For a start, students should learn to use arithmetic, branching, and simple looping statements; other techniques, such as subroutines, can be considered later as the need for them arises. Machine arithmetic can be contrasted with ordinary arithmetic, with examples of roundoff error being given.

Functions should be introduced as single-valued rules of association, with the relationships between the inputs and outputs of computer programs providing many examples of both numeric and nonnumeric functions. Questions of scaling which arise in the development of a simple program for graphing functions can be used as a bridge to the next section on linear functions.

2. Linear and quadratic functions. In studying rates of change, the student can first write a computer program to model a situation involving constant change. After this model has been used to motivate a study of linear functions, the computer program can be modified by the addition of a single statement to model constant acceleration, thereby motivating a study of quadratic functions; later, the added statement can be changed to have the program model more complicated rates of acceleration (e.g., a bouncing ball or exponential growth). Questions about zeros, maxima, and minima should be raised and answered to ascertain properties of models, functions, and graphs. In this way the study of linear and quadratic functions provides a framework for later material in the course.

In addition to using the computer for simulation, one can stress the algorithmic aspects of graphing by using programs to compute the slopes of lines or the zeros of a quadratic function by the quadratic formula. Zeros, and in particular square roots, can also be approximated by the bisection method.

3. Linear programming. The study of linear functions leads naturally to a study of linear programming in two dimensions. Boundary conditions lead to a consideration of linear inequalities and to

the solution of simultaneous equations in two unknowns in order to determine constraint regions. Cost functions can be introduced as functions from vectors in those regions to numbers, and the location of the maxima and minima of these functions at the vertices of regions can be demonstrated by drawing level curves.

4. Matrix operations. As a further example of the applicability of linear methods, models of population movement can be studied. One can introduce a vector V to represent the population distribution at a given time and a matrix M to represent the percentage redistribution of population over a year's time. Matrix and vector multiplication can be motivated by writing a computer program to model population movement over a number of years and observing that M^nV is the population distribution after n years. Finally, this model can be used to motivate the solution of simultaneous linear equations or the inversion of a matrix to find the equilibrium distribution.

5. Algebra of functions. By associating functions with subroutines which compute them, one can motivate a general discussion of the domains and ranges of functions, as well as of algebraic operations on functions. Applied to polynomials, this leads naturally to the rational functions. In order to answer standard questions about these functions, one can discuss numerical techniques such as bisection and hill-climbing for locating zeros, maxima, and minima. The inverse of a function can be found by computing the zeros of translated functions.

6. Exponential and logarithmic functions. Computations involving population growth, interest rates, or radioactive decay lead to a study of exponential functions. The logarithm can be computed by the method developed in Section 5, and its properties can be established from the properties of exponentials.

REFERENCES

No presently available text is suitable for this course. While some of the references listed below contain material that can be used in the course, no text develops computing and mathematics together along the lines suggested by units 1 and 2. The approaches to computing in two of the references bracket the suggested approach:

Vogeli, et al., is generally too elementary and does not use computing in a substantial way, while Higgins presumes too much prior experience both in computing and in mathematics. The remaining references are programming texts which contain some examples appropriate for the course.

Barrodale, Ian; Ehle, Byron; Roberts, F. D. K. Elementary Computer Applications in Science, Engineering, and Business. New York, John Wiley and Sons, Inc., 1971.

Gruenberger, Fred and Jaffray, George. Problems for Computer Solution. New York, John Wiley and Sons, Inc., 1965.

Higgins, G. Albert. The Elementary Functions: An Algorithmic Approach. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1974.

Kemeny, John G. and Kurtz, Thomas E. Basic Programming, 2nd ed. New York, John Wiley and Sons, Inc., 1971.

Maurer, H. A. and Williams, M. R. A Collection of Programming Problems and Techniques. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1972.

Vogeli, Bruce R.; Prevost, Fernand; Gilbert, Glenn; Carroll, Edward. Algebra One. Morristown, New Jersey, Silver Burdett Company, 1971.

MC-1 and MC-2. Calculus I and II with Computer Support

[Prerequisite for MC-1: MC-0 and trigonometry, or equivalent background; prerequisite for MC-2: MC-1] The introductory courses on calculus appear to be those mathematics courses in which the use of the computer has been most popular. One reason for this is the fact that many concepts and methods in the calculus have a practical flavor which can be enhanced by introducing computing. While the motivational value of computational work plays a considerable role, the student can also handle more realistic problems using the computer as a tool and with it learn to appreciate more fully the power and usefulness of calculus.

There are at this time no firm guidelines as to how the computer should be introduced into calculus courses; many radically different experiments have been conducted and are still being carried on. We believe that at present a practical and rather attractive approach is to use the computer to support courses which are more or less traditional in the selection and sequencing of the material. Hence the courses described below are, at least in outline, identical with the courses Mathematics 1 and 2 in the GCMC Commentary, and their basic purpose remains essentially the same--namely, that of being an intuitive, yet sound, introduction to limits in various forms, such as derivatives, integrals, or sums of series, along with applications of several types, such as maximum-minimum problems or questions leading to integrals.

We do not believe that it is enough to teach the calculus courses more or less as usual and to assign computer projects as supplements to the course. Such an approach does not take full advantage of the interplay between theoretical and algorithmic ideas. The new courses must be taught in a different manner, if only because computing can provide much useful motivation for the calculus. Moreover, the emphasis on what is taught should shift towards the kind of realistic applications that are possible with the computer. In these ways the computer can be used to support the presentation of material rather than merely to supplement it.

The commentaries below indicate ways in which this supportive role of the computer can be accomplished. For the most part these commentaries are meant to supplement rather than replace those in the GCMC Commentary.

COURSE OUTLINE FOR MC-1

1. Introduction. (4 hours) Review of the function concept. Function evaluation and graphing on a computer.

2. Limits, continuity. (3 hours) Limit and approximation defined intuitively. Derivatives as examples. Definition of continuity, types of discontinuity, Intermediate Value Theorem. Computational applications involving the bisection method and showing effects of truncation and roundoff error.

3. Differentiation of rational functions; maxima and minima. (5 hours) Computational projects involving search algorithms for finding extrema. Newton's method.

4. Chain rule. (3 hours) Include derivatives of functions defined implicitly, inverse function and its derivative. The algorithmic aspect of functional composition.

5. Differentiation of trigonometric functions. Higher derivatives. (3 hours)

6. Applications of differentiation. (3 hours) Tangent as "best" linear approximation. Approximations using differentials. Additional extremal problems. Related computer applications.

7. Intuitive introduction to area. (2 hours) Computational approximation of areas of regions under a curve.

8. Definite integral. (3 hours) Simple quadrature rules and their applications.

9. Indefinite integrals, Fundamental Theorem. (4 hours)

10. Logarithmic and exponential functions. (3 hours) Computer problems involving exponential growth or decay using Euler's method.

11. Applications of integration. (3 hours)

COMMENTARY ON MC-1

1. Introduction. The computer can be used to evaluate functions, thereby considerably extending the kinds of functions a student can handle and, indeed, even recognize as functions (e.g., functions with piecewise or algorithmic definitions can be evaluated numerically even though they may not be expressible algebraically). Students should recognize that the relation between the input and the output of a computer program can define a function; such an awareness can be used later to demonstrate the existence of various interesting functions.

A good computing facility would enable the student to experiment with functions--and also their graphs when they can be drawn--in much the same way as he can work with simple functions when he has only pencil and paper. A graphing program should be provided or developed, and students should become reasonably familiar with it, so that it can be used to motivate later topics in terms of graphs.

2. Limits, continuity. Many kinds of calculations help to motivate the need for a precise definition of limit. Such calculations arise in practical attempts to approximate limits of functions or rates of change. While the student will sense that successive approximations are approaching a limit, he will also discover that the limitations of numerical approximations due to truncation or roundoff errors prevent him from calculating that limit exactly. This awareness should be used to motivate the need for mathematical proofs of the existence of limits.

The bisection method for finding zeros of continuous functions can be introduced either as motivation for or as an application of the Intermediate Value Theorem.

In this section, as well as in others, one should recall several points observed earlier concerning the use or misuse of the computer. First, one should use terminology carefully so as not to

mislead students; a computer can approximate a limit, but it cannot "compute" one, nor can it "test" a function for continuity. Second, one should remember that the primary purpose of the course is still to teach calculus and that it is the algorithmic approach, and only secondarily the actual use of the computer, which advances this purpose; hence the computer does not have to be used in every conceivable situation, and many points about algorithms can be made without a computer at all.

3. Differentiation of rational functions. Nonnumerical algorithms can be recognized when they appear even though they may not be programmed. For example, formal differentiation should be recognized as a process that can be mechanized.

More realistic maximum and minimum problems can be attempted. The approach to such problems would include graphing functions, searching for extremal points, and sometimes finding zeros of derivatives. The computer increases the student's power to find zeros of functions since the bisection method or Newton's method are available when algebraic techniques fail.

4. Chain rule. Computer programs and flowcharts can be used to explain the process of functional composition and to motivate the chain rule. Information about the inverse of a function f can be obtained by finding zeros of $f(x) - a$, for various values of a.

5. Differentiation of trigonometric functions. The graphing program can be used for motivation.

6. Applications of differentiation. The limitations of numerical methods can be used to motivate the need for theorems concerning, say, the number of extremal points. For example, numerical methods may lead one to suspect that $x^2 + \cos^2(kx)$ has a unique minimum when k is slightly larger than 1, rather than two minima which are separated by a maximum at 0.

Again, more realistic maximum and minimum problems can be attempted. Newton's method for locating zeros of functions can be developed and compared with the bisection method for its rate of convergence and range of applicability.

7. Intuitive introduction to area.

8. Definite integral. The notion of the definite integral can be made concrete prior to the proof of the Fundamental Theorem so that the student need not confuse the existence of the definite integral of a function with his ability to find an antiderivative. The student can write programs to approximate definite integrals by techniques such as the trapezoidal rule. Improper integrals can be motivated in terms of programs to approximate them.

9. Indefinite integrals, Fundamental Theorem. The nature of the indefinite integral as a function of the upper endpoint can be illustrated by considering a computer program to approximate values of this function.

10. Logarithmic and exponential functions. Numerical methods can be used to discuss and sketch solutions of the differential equation $y' = ky$.

11. Applications of integration. The computer greatly increases the variety of examples which can be treated. Applications of the integral as the limit of Riemann sums, and not merely as an antiderivative, were recommended in the GCMC Commentary and can be handled much more successfully with the use of the computer. For example, in following those suggestions one can use numerical techniques to integrate the normal probability distribution or to graph a logistic curve corresponding to a differential equation $N' = (a - bN)N$ governing population growth. One can also observe the general applicability of numerical techniques as opposed to the often limited applicability of analytical techniques. For example, given experimental data concerning the acceleration of a vehicle, one can compute the values of integrals to obtain the velocity and position of that vehicle [cf. Garfunkel, Solomon. "A laboratory and computer based approach to calculus." American Mathematical Monthly, 79 (1972), pp. 282-290].

COURSE OUTLINE FOR MC-2

1. Techniques of integration. (9 hours) Integration by trigonometric substitutions and by parts; inverse trigonometric functions; quadrature formulas and computer applications; improper

integrals and numerical questions; volumes of solids of revolution.

2. Elementary differential equations. (7 hours) Elementary methods for computational solution.

3. Analytic geometry. (10 hours) Vectors; lines and planes in space; polar coordinates; parametric equations.

4. Partial derivatives. (5 hours)

5. Multiple integrals. (5 hours)

COMMENTARY ON MC-2

1. Techniques of integration. At the discretion of the instructor, less attention might be paid to techniques of formal integration in order to provide time for a study of numerical methods for approximating definite integrals. Experiments can be performed to suggest theorems about the rates of convergence of various methods. In some simple cases one might attempt to place bounds on the numerical errors due to the approximation method and to truncation and roundoff effects. In general, an applied flavor can be introduced into the calculus course by relating some of the theorems to realistic numerical processes.

Although formal integration is more complicated than formal differentiation, certain aspects, such as integration of powers of sines and cosines or the use of partial fractions, can be considered from an algorithmic point of view.

As an example of finding error bounds, consider the midpoint (or tangent) approximation

$$\int_a^b f(x)\,dx \approx h \sum_{k=1}^{n} f(a + [k - \tfrac{1}{2}]h),$$

where $h = \frac{b-a}{n}$, to the integral of a twice-differentiable function f. One can show with the aid of Taylor's theorem that the truncation error is bounded by $\frac{(b-a)h^2}{24}B$, provided that $|f''(x)| \leq B$ for $a < x < b$. Furthermore, for suitable a, b, and h, the error in evaluating the approximation is bounded by $nhE_1 + n(n-1)E_2Fh$, which is less than $(b-a)(E_1 + nE_2F)$, where E_1 is the maximum absolute error in the computation of $f(x)$ for $a < x < b$, E_2 is

a bound for $\frac{(1+r)^n - 1}{n}$ (r being the relative roundoff error bound), and $|f(x)| \leq F$ for $a < x < b$. In the particular case $n = 128$ and $E_1 = E_2 = 1.01 \times 2^{-27}$, the midpoint approximation to $\int_1^2 \frac{dx}{x} = \log 2$ can be guaranteed to have an error of no more than 10^{-5}.

It should be observed that formal and numerical methods are not mutually exclusive alternatives, and that many problems require a combination of the two. Analytical techniques may be used to transform an integral for numerical methods. For example, the integral

$$\int_{\frac{1}{2}\pi}^{\infty} \frac{\sin x}{x}\, dx$$

is more easily handled numerically if it is first transformed to

$$\frac{4}{\pi^2} - 2\int_{\frac{1}{2}\pi}^{\infty} \frac{\sin x}{x^3}\, dx,$$

which is obtained by integrating by parts twice.

2. Elementary differential equations. The notion of a tangent field can be used to suggest numerical methods for the approximate solution of first-order differential equations. Higher-order equations can also be treated by translating them into systems of first-order equations which can then be solved numerically.

A bound on the propagated error for a simple method can be derived. With Euler's method applied to $y' = f(x,y)$, $y(x_o) = y_o$, it can be shown that the propagated error is bounded by

$$\frac{R + T}{hL} \exp[L(x_n - x_o)],$$

where R is a bound on the local "roundoff" error

$$y_n^c - y_{n-1}^c - hf(x_{n-1}, y_{n-1}^c),$$

T is a bound on the local "truncation" error

$$y(x_n) - y(x_{n-1}) - hf(x_{n-1}, y(x_{n-1})),$$

h is the step-size, L is a Lipschitz constant, and y_n^c is the computer approximation to $y(x_n)$, $x_n > x_o$. [See Gear, C. William. Numerical Initial Value Problems in Ordinary Differential Equations. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1971.]

3, 4, 5. Analytic geometry, Partial derivatives, Multiple integrals. Due to the lack of numerical methods which are both elementary and practical, the computer itself has less impact on the teaching of multivariable calculus than on single-variable calculus. However, an algorithmic approach can still be used, for example, to stress analogies with single-variable calculus or to introduce formal manipulations. If computing applications are desired, one might discuss hill-climbing techniques for finding maxima or some relatively simple method for approximating the values of double integrals.

REFERENCES

The following texts contain elementary applications of numerical methods to the calculus.

1. Sources of applications

Barrodale, Ian; Ehle, Byron L.; Roberts, F. D. K. Elementary Computer Applications in Science, Engineering, and Business. New York, John Wiley and Sons, Inc., 1971.

Dorn, William S.; Bitter, Gary G.; Hector, David L. Computer Applications for Calculus. Boston, Massachusetts, Prindle, Weber, Inc., 1972.

Hamming, Richard W. Calculus and the Computer Revolution. Boston, Massachusetts, Houghton Mifflin Company, 1968.

Hull, Thomas E. and Day, David D. F. Computers and Problem Solving. Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1969. Chapter 9.

2. Some calculus texts having a computational flavor

Flanders, Harley; Korfage, Robert R.; Price, Justin J. Calculus. New York, Academic Press, Inc., 1970.

Henriksen, M. and Lees, M. Single-Variable Calculus. New York, Worth Publishers, Inc., 1970.

Stenberg, Warren, et al. Calculus, A Computer Oriented Presentation, Parts 1 and 2. CRICISAM, Florida State University, 1970.

MC-DM. Discrete Mathematics

[Prerequisite: The material of this course can be taught at various levels of difficulty and sophistication in the undergraduate curriculum. Although no specific college mathematics courses are prerequisite for MC-DM, it is important that the student have ability in manipulating symbols and in using formulas.] Although an obvious goal of this course is to equip the students with some useful mathematical tools, a more important goal is to develop their ability to perceive, formulate, and solve problems that are discrete in nature. This course can be taken prior to, or concurrently with, a course in calculus. Indeed, students who are not mathematics majors might be counseled to take such a course rather than the traditional freshman calculus course. (For example, it can be argued that for social science, behavioral science, and biological science majors, a course in discrete mathematics might be more useful than a course in calculus.) There are several undergraduate-level mathematical courses that can follow this course naturally. A course in Probability and Statistics (see, for example, 4.2 below) or a course in Applied Algebra may well be popular choices. Other courses are Combinatorial Mathematics, Optimization Techniques (such as those proposed by the Panel on Applied Mathematics in the CUPM report Applied Mathematics in the Undergraduate Curriculum), Applied Logic, Graph Theory, and Computational Algorithms. Although computing facilities are not absolutely essential in such a course, they can play a very attractive supporting role. Since there is a strong algorithmic flavor throughout this course, the implementation of some computational algorithms will enhance the understanding and appreciation of the mathematical results. Also, probably in a less significant way, ideas such as graphical representation of discrete functions and solution of difference equations can be illustrated on a computer.

The number of hours specified is intended to indicate the relative emphasis for the various topics. Some instructors will find these time estimates unsuitable and will therefore need to make adjustments for their classes. However, because the material covered in this course is so new and unusual in the undergraduate curriculum, the Panel felt it would be valuable to present a wide variety of ideas for a course in discrete mathematics.

COURSE OUTLINE

1. Elementary set theory. (2 hours) Basic concepts and terminology in set theory. Subsets. Empty set. Intersection, union, symmetric difference, and complementation of sets. Venn diagrams.

2. Permutations and combinations. (4 hours) Permutation and combination of objects. Simple enumeration formulas such as that for the number of ways to select or to arrange r objects from n

objects with or without repetitions. Simple machine tools of combinatorics such as computer algorithms for generating all permutations and all combinations of a set of objects.

3. Discrete functions. (2 hours) Domain and range of a function. One-to-one and onto functions. Pigeonhole principle.

4. Manipulation of discrete functions. (2 hours) Forward and backward differences of a discrete function. Accumulated sum of a discrete function. Sum, product, convolution, and correlation of discrete functions.

5. Generating functions. (4 hours) Generating functions as alternative representations of discrete functions. Operations on discrete functions and the corresponding operations on their generating functions.

6. Difference equations. (5 hours) Linear difference equations with constant coefficients. Homogeneous solution and particular solution. Boundary conditions and undetermined coefficients. Solution of difference equations by the technique of generating functions. Simultaneous difference equations.

7. Relations. (2 hours) Cartesian product of sets. Binary relations. Reflexive, symmetric, transitive relations. Equivalence relations. Partial ordering relations. Union, intersection, and complementation of relations.

8. Graphs. (2 hours) Basic terminology in the theory of graphs. Directed graphs. Linear graphs and multigraphs. Connectedness. Paths. Graphs as representations of binary relations. Graphs as structural models.

9. Trees, circuits, and cut-sets. (4 hours) Mathematical properties of trees. Trees as structural models. Spanning trees. Circuits. Cut-sets.

10. Path problems in graphs. (5 hours) Eulerian path. Hamiltonian path. Existence of Eulerian paths and Hamiltonian paths in graphs. Physical interpretation of these notions. Shortest path algorithms and related problems.

11. Network flow problems. (4 hours) Transportation networks. Maximum-flow minimum-cut theorem. The Ford-Fulkerson algorithm for finding maximum flow.

1. Elementary set theory. The theme of this course is "discrete objects and their relationships." Consequently, the language of elementary set theory will be used throughout the course. The discussion should remain at an intuitive level, although it is quite reasonable to mention topics such as Russell's paradox which may lead to a discussion of axiomatic set theory.

2. Permutations and combinations. The discussion can begin with a determination of the number of subsets of a given set, a natural continuation of the material in Section 1. An important lesson to teach the students is that often some seemingly difficult problems may have very simple methods of solution when considered from the correct point of view.

Example: Design an algorithm for generating all r-combinations of n objects with unlimited repetitions.

Example: From all 5-digit numbers a number is selected at random. What is the probability that the number selected has its digits arranged in nondescending order?

[Answer: $\binom{10 + 5 - 1}{5}\ 10^5$]

3. Discrete functions. The notion of discrete functions is introduced as an association of values (elements in the range) to objects (elements in the domain). There are numerous examples of discrete functions: coloring the faces of a polyhedron, assigning grades to students, classification of documents, etc. Point out the obvious extension to the notion of continuous function. The pigeonhole principle (also known as the shoebox argument) is a powerful technique, although it sounds extremely simple, as the following example illustrates.

Example: The integers 1, 2, 3, ..., 101 are arranged randomly in a sequence. Show that there is either a monotonically increasing subsequence or a monotonically decreasing subsequence of 11 (or more) integers.

[Solution: Let $a_1, a_2, a_3, \ldots, a_{101}$ denote a random arrangement of the integers 1, 2, 3, ..., 101. Let us label each

integer a_k with a pair of numbers (i_k, j_k), where i_k is the length of a longest monotonically increasing subsequence that begins at a_k, and j_k is the length of a longest monotonically decreasing subsequence that begins at a_k. Suppose that $1 \leq i_k \leq 10$, $1 \leq j_k \leq 10$ for $k = 1, 2, \ldots, 101$. According to the pigeonhole principle, there must exist a_m and a_n which are labelled with the same pair of numbers. However, this is an impossibility because $a_m < a_n$ implies that $i_m > i_n$, and $a_m > a_n$ implies that $j_m > j_n$. (We assume that $m < n$.)]

4. Manipulation of discrete functions. The notion of the forward and backward differences of a discrete function corresponds to the notion of the derivative of a continuous function. The notion of the accumulated sum of a discrete function corresponds to the notion of the integral of a continuous function. The convolution $z(n)$ of two discrete functions $x(n)$ and $y(n)$ is defined to be

$$z(n) = \sum_i x(i)\, y(n-i).$$

The crosscorrelation function $w(n)$ of two discrete functions $x(n)$ and $y(n)$ is defined to be

$$w(n) = \sum_i x(i)\, y(i-n).$$

The autocorrelation of a function is the crosscorrelation of the function with itself.

Example: Consider the sequence $A = \{1, 1, 1, -1, 1, 1, -1\}$ as a signal transmitted by a radar transmitter. This signal is bounced back by an object whose distance from the radar is to be measured. (The distance can be determined from the elapsed time between the transmission of the signal and the arrival of the return signal.) To minimize the effect of noise interference, we want to choose a sequence so that the correlation function between the transmitted and the received signals will have a large peak value. Show that A is a good choice. [Answer: The autocorrelation function of the sequence A is $\{-1,0,-1,0,-1,0,7,0,-1,0,-1,0,-1\}$, which has a large peak value.]

5. Generating functions. The concept of the generating function of a discrete function corresponds to the concept of the Laplace transformation or the Fourier transformation of a continuous function. The sum of two discrete functions corresponds to the sum of their generating functions. The convolution of two discrete functions corresponds to the product of their generating functions.

Example: Show that

$$\binom{n}{0}^2 + \binom{n}{1}^2 + \binom{n}{2}^2 + \ldots + \binom{n}{n}^2 = \binom{2n}{n}.$$

Give a combinatorial interpretation (in terms of selection of objects) of this equality. [Answer: When a coin is tossed $2n$ times, there are 2^{2n} sequences of possible outcomes. Both sides of the above equality give the number of sequences of outcomes in which the number of heads occurring in the first n tosses is equal to the number of heads occurring in the last n tosses.]

6. Difference equations. Students will be better prepared for a course in differential equations after they have studied difference equations in this course. Indeed, concepts such as homogeneous solutions and particular solutions carry over directly to differential equations. Solving difference equations by the technique of generating functions corresponds to solving differential equations by the technique of Laplace transformations.

Example: A certain nuclear reaction in a system containing nuclei and high and low energy free particles is described as follows. There are two kinds of events: (i) a high energy particle strikes a nucleus, causing it to emit 3 high energy particles and 1 low energy particle, and is absorbed; (ii) a low energy particle strikes a nucleus, causing it to emit 2 high energy particles and 1 low energy particle, and is absorbed. Every free particle causes an event 1 μsec after it is emitted. If a single high energy particle is injected at time $t = 0$ into a system containing only nuclei, what will the total number of free particles in the system be at time $t = 20$ μsec? [Solution: Let a_n

denote the number of high energy particles and b_n the number of low energy particles in the system at the n^{th} μsecond. We have the simultaneous difference equations: $a_n = 3a_{n-1} + 2b_{n-1}$ and $b_n = a_{n-1} + b_{n-1}$, with the initial conditions $a_o = 1$ and $b_o = 0$. Solving these equations, we obtain $a_n + b_n = \frac{1}{2\sqrt{3}} \{(2 + \sqrt{3})^{n+1} - (2 - \sqrt{3})^{n+1}\}$.]

7. Relations. Structural properties of sets of discrete objects can be described by relations. There are numerous examples of the concept of relations between objects: for instance, the relation "is the father of" is nonreflexive, nonsymmetric, and nontransitive; the relation "is the spouse of" is symmetric; the relation "is divisible by" (between integers) is a partial ordering relation.

Example: Write a computer program to determine all possible assignments of 0's and 1's to the vertices of the partial ordering diagram in Fig. 1 so that a 1 never precedes a 0.

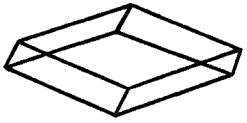


Fig. 1

8. Graphs. There are many examples from various disciplines using graphs as abstract models of structures, among which are social structures, finite state machines, PERT charts, data structures in computer programs.

Example: The inputs to an electronic combination lock are strings of 0's and 1's. The lock will be opened when the pattern 010010 appears at the end of the input string. Such a lock can be modeled graphically as in Fig. 2, where a string of 0's and 1's defines a path starting at the initial vertex.

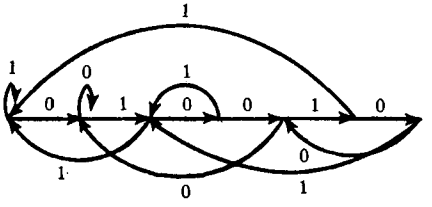


Fig. 2

9. Trees, circuits, and cut-sets. Although trees are very simple in concept, they are rich in structure and find application in many areas of study. There is enormous room for further discussion beyond the basic concepts; topics such as enumeration of trees, optimal trees (notably the Huffman algorithm for determining trees with minimum weighted path lengths), and algorithms for traversing trees may be considered.

Example: Communication links are to be built between cities. Suppose the cost of building a link between two cities is proportional to the distance between them. We want to build a set of links so that there is a path through these links between every two cities. Design a nonexhaustive algorithm that will yield a layout of minimal total cost. (This is a problem of designing an algorithm for finding a minimal spanning tree in a graph with weighted edges.)

10. Path problems in graphs. The notion of a shortest path in a graph has a clear interpretation in physical terms. If computing facilities are available, the implementation of some graph algorithms by students would be highly desirable. The discussion of Eulerian paths brings out another feature in our study of discrete structures--a simple criterion for the existence of some properties in a large class of structures. The following examples also illustrate some physical interpretations of the abstract notions of Eulerian and Hamiltonian paths.

Example: Arrange all n-digit binary numbers in such a way that the last (n-1) digits of a number are equal to the first (n-1) digits of the successive number. (This is an Eulerian path problem with application in digital engineering.)

Example: Arrange all n-digit binary numbers in such a way that two adjacent numbers differ only at one digit. (This is a Hamiltonian path problem with application in digital engineering.)

11. Network flow problems. This discussion not only exposes the students to the general problem of discrete optimization but also shows them a recursive technique in which the solution is improved in a step-by-step manner until an optimal solution is reached. This has exactly the same flavor as that of the simplex method in linear programming.

Example: Engineers and technicians are to be hired by a company to participate in three projects. The personnel requirements of these three projects are listed in the following table:

Minimal number of people needed in each project		Minimal number in each category			
		Mechanical engineers	Mechanical technicians	Electrical engineers	Electrica technicia
Project I	40	5	10	10	5
Project II	40	10	5	15	5
Project III	20	5	0	10	5

Moreover, to prepare for later expansion, the company wants to hire at least 30 mechanical engineers, 20 mechanical technicians, 20 electrical engineers, and 20 electrical technicians. What is a minimal number of persons in each category that the company should hire, and how should they be allocated to the three projects? (This problem can be formulated as a problem of finding a minimal flow in a transportation network where there is a lower bound on the flow-value in each of the edges.)

REFERENCES

The following four books would be useful in a course at the freshman-sophomore level. Berman and Fryer contains a very broad coverage of combinatorics. Kemeny, Snell, and Thompson discusses many interesting application problems. Berztiss was written mainly for computer science students. Vilenkin has a nice collection of examples and problems.

Berman, Gerald and Fryer, K. D. Introduction to Combinatorics. New York, Academic Press, Inc., 1972.

Berztiss, A. T. Data Structures. New York, Academic Press, Inc., 1972.

Kemeny, John G.; Snell, J. Laurie; Thompson, Gerald L. Introduction to Finite Mathematics, 2nd ed. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1966.

Vilenkin, N. Y. Combinatorics. New York, Academic Press, Inc., 1971.

The following books are more suitable for a junior-senior level course. They can also be used as references in a freshman-sophomore level course. Although these books are more advanced than the books cited above, they are quite readable even for undergraduate students.

Berge, Claude. Principles of Combinatorics. New York, Academic Press, Inc., 1971.

Berge, Claude. The Theory of Graphs and its Applications. New York, John Wiley and Sons, Inc., 1962.

Busacker, Robert G. and Saaty, Thomas L. Finite Graphs and Networks: An Introduction with Applications. New York, McGraw-Hill Book Company, 1965.

Knuth, Donald E. The Art of Computer Programming, vol. I. Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1968.

Liu, C. L. Introduction to Combinatorial Mathematics. New York, McGraw-Hill Book Company, 1968.

Ryser, Herbert J. Combinatorial Mathematics, MAA Carus Monograph 14. New York, John Wiley and Sons, Inc., 1963.

MC-3. Algorithmic Elementary Linear Algebra.

[Prerequisite: MC-0 or equivalent background] This course corresponds to Mathematics 3, "Elementary Linear Algebra," as described in the GCMC Commentary, and we refer the reader to that report for some additional comments. The differences between the two courses are mainly matters of emphasis and arrangement of topics. Whereas the course described in the GCMC Commentary stresses the algebraic and geometrical aspects of linear algebra and has a certain abstract flavor, the present course has a predominantly algorithmic viewpoint and its discussion revolves around the various ramifications of solving a system of linear equations. Throughout the course, detailed algorithms are to be presented and discussed, in flowchart or some simple step-by-step form, and the students should use these in connection with various practical problems, on a computer where possible. At the same time, in this course it is particularly important to warn the student that the algorithms are based on arithmetic with real numbers and that in a practical computation the effect of roundoff errors may lead to considerable distortions of the final result. This may be illustrated with well-chosen examples, but no attempt should be made to enter into a deeper discussion of such numerical problems.

COURSE OUTLINE

1. Introduction. (3 hours) Discussion of various practical problems involving matrices. Review of the elimination process for 2 X 2 and 3 X 3 systems of equations. Examples showing various cases of solvability of such systems.

2. Matrix algebra. (5 hours) Definition of real n X m matrices. Examples of various special forms of matrices. Transposes; symmetric and diagonal matrices. Equality, addition, and scalar multiplication. Matrix product and its properties. Computational applications of matrix algebra.

3. Vectors and geometry. (4 hours) Geometrical interpretation of 1 X 3 matrices. Algebraic properties in 3-space. The inner product. Euclidean length, angle, orthogonality, direction cosines. Linear combinations. Lines and planes. Projections. Vector proofs of simple geometric theorems. Matrices as linear transformations in R^2 and R^3. Geometric interpretation of one linear equation in three variables and of a 3 X 3 system.

4. Inverses and the row echelon form. (5 hours) Left and right inverses of an n X m matrix and relation to existence and

uniqueness of solutions of linear equations. Review of the elimination process as motivation for the elementary row operations. Elementary row operations and their formulation in terms of multiplication by elementary matrices. The algorithm for transforming a matrix to row echelon form. Equivalent systems of equations. Solvability properties of homogeneous and inhomogeneous systems using row echelon form.

5. Linear dependence and independence. (5 hours) Linear combinations of n × m matrices. Linear spaces of vectors and matrices. Subspaces. Linear dependence and independence of vectors in R^3 and in R^n, and of matrices. Examples and basic properties. Use of row echelon form to determine linear dependence or independence in R^n. Bases. Exchange algorithm. Dimension. Sum and intersection of subspaces and their dimensions.

6. Elimination. (5 hours) Algorithm for solving triangular systems. Inverses of triangular matrices. Gaussian elimination without pivoting; triangular decomposition. Pivots and the general algorithm. Backsubstitution and the solution of square linear systems. Algorithm for the computation of inverses. Numerical examples of ill-conditioning.

7. Rank. (5 hours) Linear mappings between linear spaces. Range space and null space. Relation between algebra of mappings and of matrices. Uniqueness aspect of row echelon form. Rank of a matrix. Rank of the transpose. Dimensions of null space and range space and related results.

8. Euclidean spaces. (4 hours) The inner product. Schwarz inequality. Euclidean length in R^n. Orthogonal bases. Gram-Schmidt process. Orthogonal projections. The least squares method.

9. [Optional] Abstract vector spaces. Axiomatic definition of vector space over R. Examples. Linear transformations and their algebra. The matrix of a linear transformation with respect to a given basis. Change of basis.

1. Introduction. Practical problems involving matrices abound. They may include the adjacency matrix of a street net, a simple resistive electrical network, a Markov chain example, the method of least squares, etc. [See, e.g., Noble, Ben. Applied Linear Algebra. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1969, Chapter 2.]

2. Matrix algebra. The stress here is on the algorithms of matrix algebra. Matrix multiplication can be motivated by practical examples of inner products leading to the product of a 1 × n matrix by an n × 1 matrix. Then the transformation of variables in linear equations readily provides a motivation of the matrix product. The examples introduced earlier can now be elaborated; for example, connectivity of a street net can be determined by forming powers of the adjacency matrix. A subroutine package for matrix algebra may be very useful for these applications.

3. Vectors and geometry. This section is rather standard. For comments we refer the reader to the GCMC Commentary.

4. Inverses and the row echelon form. In this section a basic algorithm is introduced, namely, the reduction to row echelon form; it will play a central role in the remainder of the course. Various applications are possible--for instance, determining solvability properties of a resistive electrical network.

5. Linear dependence and independence. In discussing the use of the row echelon form to determine linear dependence and independence, it is particularly important to illustrate the numerical problems which might occur when a computer is used. This can be motivated well by simple 2- and 3-dimensional examples. If time permits, the role of the exchange algorithms in linear programming can be illustrated by simple examples. [See, e.g., Stiefel, E. L. Introduction to Numerical Mathematics, translated by W. C. Rheinboldt. New York, Academic Press, Inc., 1963.]

6. Elimination. After a thorough discussion of the overall algorithm, it may be desirable to use a well-written subroutine package for computer assignments involving the solution of linear

systems arising in the practical problems introduced earlier. [See, for example, the routines given in Forsythe, George E. and Moler, C. Computer Solution of Linear Algebraic Systems. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1967.]

7. Rank. For some applications of rank, e.g., to chemical reactions, see Chapter 5 of the book by Ben Noble which was cited above.

8. Euclidean spaces. Again, applications abound. In particular, various problems leading to the use of the least squares method can be discussed.

REFERENCES

1. Matrices and linear algebra

Davis, Philip J. The Mathematics of Matrices. Waltham, Massachusetts, Blaisdell Publishing Company, 1965. A well-written elementary introduction to matrices.

Hohn, Franz E. Elementary Matrix Algebra, 2nd ed. New York, The Macmillan Company, 1964. An introductory text which proceeds in a manner similar to the outline above.

Noble, Ben. Applied Linear Algebra. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1969. This excellent text corresponds in spirit and approach to our outline but contains considerably more material and is, in parts, somewhat more advanced.

2. Numerical aspects

Forsythe, George E. and Moler, C. Computer Solution of Linear Algebraic Systems. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1967. A brief, modern, but more advanced text on the basic numerical aspects of solving systems of linear equations.

Fox, Leslie. Introduction to Numerical Linear Algebra. New York, Oxford University Press, Inc., 1965. This text is a good source of instructive examples of error problems in numerical linear algebra.

4. Further Undergraduate Courses

In this section we discuss a rather heterogeneous group of upper-division undergraduate mathematics courses and areas affected by computing. The given list does not exhaust the possibilities and, even for the areas discussed here, there may well be other ways of incorporating the effect of computing. Clearly, at this level there is considerably more flexibility and there are probably many ways of modifying the approaches we suggest here.

For the courses in this section, the programming prerequisites are, of course, more advanced than for the previous courses; the computational facilities may also need to be more flexible. (See Sections 5.1 and 5.2). Further, the knowledge of computing and applied mathematics required by faculty members teaching these courses differs considerably from course to course. Thus for Ordinary Differential Equations (4.1) and Numerical Calculus (4.4) a knowledge of numerical analysis as well as facility in programming are absolutely essential. For Discrete Probability and Computing (4.2) reasonable programming experience in addition to a knowledge of probability is required. For Algebra Courses Influenced by Computing (4.5) a grounding in the algebraic foundations of computer sciences is needed in addition to the more usual kinds of computer expertise. Finally, for Mathematical Computer Modeling (4.3) a thorough knowledge of the applications involved is essential, of course, in addition to the programming and numerical analysis knowledge required by the selected applications.

4.1 Ordinary Differential Equations (3 semester hours)

[Prerequisites: MC-3, Mathematics 4 from the GCMC Commentary, good programming experience] How ordinary differential equations arise in practice. Separation of variables, integrating factors, variation of parameters, substitution. Equations with constant coefficients. Series solutions. Euler's method and a brief treatment of existence and uniqueness. An explicit Runge-Kutta method; a trapezoidal method for stiff systems. [For a discussion of stiff systems, see Gear, C. William. Numerical Initial Value Problems in Ordinary Differential Equations. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1971.] An introduction to boundary value problems.

The purpose of this course is basically the same as that of a more traditional course on ordinary differential equations, except that greater emphasis should be given to practical methods of solution. The most significant change is the inclusion of several carefully chosen numerical methods.

One numerical method is based on a well-established Runge-Kutta formula and is treated in enough detail to permit the writing of a reasonably effective computer program. This method is adequate for

nonstiff problems, and there is no need to make more than brief reference to more complicated methods, such as multi-step methods, for these problems. However, one other method is needed for stiff systems of ordinary differential equations. A method based on the trapezoidal rule is included in this course because it is both simple and adequate. Numerical methods for boundary value problems are also included. A more detailed discussion of other numerical methods should be left to courses in numerical analysis.

This course can be followed by a second semester course covering several more advanced topics and exploiting more thoroughly various numerical methods. Topics for such a second semester may be chosen from among the following:

> Series solutions (including special functions), autonomous systems, Laplace transform, comparison theorem, eigenvalues and eigenfunctions, perturbation theory, asymptotic behavior, numerical methods, Galerkin methods, applications.

COURSE OUTLINE

1. Systems of equations. (2 hours) How ordinary differential equations arise in physical, chemical, biological, and economic problems.

2. Elementary analytic methods. (5 hours) Variables separable, e.g., in $y' = 1 - y^2$. Integrating factors, e.g., in $y' + P(x)y = Q(x)$. Substitution, e.g., in $y' = (ax + by)/(cx + dy)$. Variation of parameters. Solving equations with constant coefficients, e.g., the system $y' = ay + bz$, $z' = cy + dz$, or the higher-order equation $y'' + ay' + by = f(x)$. Introduction to series solutions.

3. Euler's method. (5 hours) Brief treatment of an existence and uniqueness theorem (perhaps without a detailed proof) of the Cauchy-Lipschitz kind, which can also be viewed as a theorem about the convergence of a simple numerical procedure. Bound on propagated error with Euler's method. Numerical examples, including a difficult one such as the Volterra equations that often arise in biological problems, e.g., $y' = 2(y - yz)$, $z' = -z + yz$.

4. More efficient numerical methods. (6 hours) Motivation of explicit Runge-Kutta formulas. A complete numerical method, including a strategy for changing step-size (see flowchart given below). Numerical examples, comparison with Euler's method. Note

the generality of the numerical method for systems of first-order equations: it can be used with nonlinear as well as linear equations; moreover, higher-order equations can be reduced to systems of first-order equations. Brief mention of more complicated multi-step methods.

5. Stiff systems of equations. (6 hours) The inability of standard methods to cope efficiently with stiff systems (e.g., with stable linear systems whose eigenvalues differ in magnitude by large factors). A complete numerical method for stiff systems based on the trapezoidal rule. Numerical examples, e.g., $y' = -101y - 100z$, $z' = y$. Compare Runge-Kutta and trapezoidal methods. Brief mention of other methods for stiff systems.

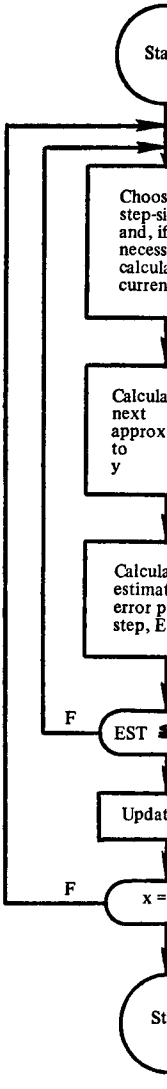
6. Boundary value problems. (10 hours) Elementary theoretical considerations, including an introduction to eigenvalues and eigenfunctions. Shooting methods. Finite difference methods. Mention of Galerkin methods.

7. Limitations of numerical methods. (2 hours) Acknowledge the limitations of numerical methods and the need for their improvement. Point out the need for further analysis of solutions of differential equations, for example in the neighborhood of a singularity.

COMMENTARY

This course is intended to provide a reasonable balance between analytic and numerical methods that can be applied to problems involving ordinary differential equations. The students are expected to carry out numerical work related to applications.

This theme can be illustrated with Volterra's equations, which are mentioned above in the detailed outline. To begin with, examples of this sort are easily motivated in terms of predator-prey relationships. Then analytic methods can provide some useful information, such as existence and uniqueness of the solutions, and, with certain initial conditions, the existence of periodic solutions. But finding reasonable approximations to the solutions involves the use of numerical methods. The analytic methods are limited to relatively simple problems but help to provide an understanding for more general situations. The numerical methods are much more generally applicable, but



Start with f (x,y), x_0, y_0, x_f (final value of x), τ (tolerance per unit step), h_{max} (maximum step-size)

$$h = \begin{cases} \min\{h_{max}, x_f - x\}, & \text{on entry} \\ \min\left\{h_{max}, x_f - x, .9\left(\frac{\tau}{EST}\right)^{1/4} h_{old}\right\}, & \text{otherwise} \end{cases}$$

Find slope on entry and after successful step

Use Kutta's formula

$[y_{i+1} = y_i + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3)$,

where $k_0 = hf(x_i, y_i)$

$k_1 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_0)$, etc.] twice with step-size $\frac{1}{2}h$

Find approximation $\bar{y}$ by using Kutta's formula once with step-size h; then EST = $\left|\frac{y - \bar{y}}{15h}\right|$

Stop with x_0, y_0 replaced by x_f and the computed approximation to $y(x_f)$

they do not contribute very much to one's understanding; moreover, it is often difficult to assess their reliability.

We include in this section a flowchart for the explicit Runge-Kutta method and some comments on a trapezoidal method for stiff systems.

Notes:

1. Choosing h to be $.9(\tau/EST)^{\frac{1}{4}}$ times its previous value can be justified as follows. First of all, the exponent is $\frac{1}{4}$ because the method is a fourth-order method and the ratio $(\tau/EST)^{\frac{1}{4}}$ is asymptotically equal to the ratio of step-sizes associated with errors of τ and EST respectively. The trial step-size should be chosen to be somewhat smaller than what is determined by this ratio, and the factor .9 has been shown experimentally to be reasonably good.

2. Some modifications of the above are needed if we wish to allow $x_f < x_o$.

3. Care should be taken to avoid possible overflow in calculating τ/EST.

4. Provision could be made for using an error exit if the error test fails with h equal to a given h_{min}.

A trapezoidal method for stiff systems

A relatively simple method for stiff systems can be patterned on the flowchart given above. The only major change that needs to be made is to replace Kutta's formula with the trapezoidal formula

$$y_{i+1} = y_i + \frac{h}{2}(y_i' + y_{i+1}')$$

and to arrange for this equation to be solved by Newton's method. (The latter is required because simple iterations on this formula will not usually converge for stiff systems.)

Some minor changes are also needed. The exponent $\frac{1}{4}$ which is used in finding h must be replaced by $\frac{1}{2}$ because the trapezoidal formula is only second-order, and the factor 15 in the formula for EST must be replaced by 3 for the same reason.

REFERENCES

There is no one book which contains all the topics described in this outline. However, the following books taken together cover the material, although the last three especially contain too much for this one course; thus, topics will have to be selected.

Boyce, William E. and DiPrima, Richard C. Elementary Differential Equations and Boundary Value Problems, 2nd ed. New York, John Wiley and Sons, Inc., 1969.

Coddington, Earl A. An Introduction to Ordinary Differential Equations. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1964. Complete coverage of the theoretical aspects.

Davis, Harold T. Introduction to Nonlinear Differential and Integral Equations. New York, Dover Publications, Inc., 1960. Good treatment of practical examples, including the Volterra equations.

Gear, C. William. Numerical Initial Value Problems in Ordinary Differential Equations. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1971. Numerical methods for initial value problems, including stiff systems.

Keller, Herbert B. Numerical Methods for Two-Point Boundary-Value Problems. Boston, Massachusetts, Ginn and Company, 1968.

4.2 Discrete Probability and Computing

[Prerequisites: MC-2 and some knowledge of programming and computing procedures such as those found in Sections 2 and 3 of MC-DM] This course is intended as an introduction to the elements of probability. The main difference between it and a standard probability course, apart from the use of computing, is that, in order to get to more complex problems, less time is spent developing tools for solving simple problems. This difference is reflected in the amount of time allotted to the various units comprising the course, as well as in the fact that difficult theorems (such as the Central Limit Theorem) are to be stated without proof. However, in cases where proofs are omitted, the computer is used to provide experimental intuition for the validity of the theorems.

COURSE OUTLINE

1. Definition of a discrete probability measure; conditional probability for experiments with a finite number of outcomes. (3 hours)

2. The frequency concept of probability; fluctuation theory

illustrated by simulation; the arcsine law for the number of times in the lead. (3 hours)

3. Sums of sequences of independent random variables with common distribution; generating functions; mean; variance. (7 hours) Computational illustration of the Central Limit Thoerem. Proof of the Weak Law of Large Numbers. Illustration of the Strong Law of Large Numbers by simulation.

4. Brief discussion of probabilities on infinite spaces. (4 hours) Poisson, normal, and exponential distributions. Waiting-time problems illustrated by computer simulation.

5. Fair games (martingales). (6 hours) System theorems. Ruin probabilities. The meaning of convergence of nonnegative martingales illustrated using branching processes and other examples.

6. Finite Markov chains. (6 hours) Recurrent and absorbing chains. Use of matrix computation to write programs for basic descriptive quantities relating to Markov chains.

7. Additional topics. (7 hours) Applications of previous topics to selected problems in discrete potential theory, simulation of complex systems, or statistics.

COMMENTARY

1. Definition of a discrete probability measure. This unit represents in part a survey of material from MC-DM. Counting is restricted to permutations and combinations. Computational applications involve the properties of the binomial coefficients.

2. The frequency concept of probability. A possible computer assignment involves the discovery of the highly unintuitive arcsine law for the number of times in the lead in a penny-matching game. Once a conjecture has been established on the basis of experiments, a proof can be given using Feller's treatment based on the reflection principle. This provides an example of an easy limit theorem.

3. Sums of sequences of independent random variables with common distribution. Let $X_1, X_2, \ldots$ be a sequence of independent integer-valued random variables, and let $S_n = X_1 + \ldots + X_n$. A computer program can be used to compute

$$p_j^{(n)} = Pr[S_n = j]$$

using only that

$$p_j^{(n)} = \sum_k p_k^{(1)} \, p_{j-k}^{(n-1)}.$$

This program may then be employed to motivate the concepts of mean and variance and to illustrate the Central Limit Theorem.

4. Brief discussion of probabilities on infinite spaces. This unit is included primarily to provide background for a later course on statistics. The discussion should be limited to distributions for a single experiment, with concepts such as the mean and variance being introduced by analogy with the finite case.

5. Fair games (martingales). Chapter 8 of Kemeny, Schleifer, Snell, and Thompson provides source material. This unit could be replaced by a unit on branching processes and generating functions.

6. Finite Markov chains. For a discussion of some related computational work, see, for example, Kemeny, John G. and Kurtz, Thomas E. Basic Programming, 2nd ed. New York, John Wiley and Sons, Inc., 1971, especially Section 16.3.

7. Additional topics. The purpose of this unit is to unify numerous applications through techniques discussed previously. For example, discrete potential theory can be applied to optimal stopping problems and to Markov decision processes; and the solution of the Dirichlet problem can be found using (a) the voltage in an electrical network, (b) the value of a stopped martingale (the Monte Carlo method), and (c) Markov chain methods. Such applications would build upon units 5 or 6 or both. Alternative or additional applications could include the simulation of complex systems (cf. Forester) or an introduction to elementary statistics.

REFERENCES

There is no single text which is suitable for the entire course. Sections of Feller and of Kemeny, Schleifer, Snell, and Thompson can be used for various units of the mathematical topics, i.e., noncomputational aspects. Freiberger gives a more advanced treatment, and Forester is an example of an application of these ideas to a real-life problem.

1. Mathematical background

Feller, William. An Introduction to Probability Theory and Its Applications, vol. 1, 3rd ed. New York, John Wiley and Sons, Inc., 1968.

Kemeny, John G.; Schleifer, A.; Snell, J. Laurie; Thompson, Gerald L. Finite Mathematics with Business Applications, 2nd ed. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1972.

2. Some computational applications

Forester, Jay W. Urban Dynamics. Cambridge, Massachusetts, MIT Press, 1969.

Freiberger, Walter F. and Grenander, Ulf. A Short Course in Computational Probability and Statistics. New York, Springer-Verlag New York, Inc., 1971.

4.3 Experimental Development of a Course in Mathematical-Computer Modeling

A mathematical model of a phenomenon, mechanism, or process can be a system of algebraic, differential, difference, or functional equations, a stochastic process, or an abstract structure in terms of which a problem or question can be studied and can be given a mathematical solution. The usefulness of mathematical models in the physical sciences and engineering is beyond question; in many instances the models are so good that computer simulation is as accurate as any experimental measurements that can be made. The power of the computer to simulate and to compute widens the scope of acceptable models, affects the usefulness of mathematical methods, and makes possible procedures which are much different from those of the past and far superior to them.

In view of the complexity of physical phenomena which have been successfully subjected to mathematical analysis, mathematicians and scientists do not doubt that useful mathematical models can be constructed in all of the sciences. Indeed, for a long time we have witnessed a growing mathematization within the nonphysical sciences.

In all of this we are just beginning to appreciate the impact of the computer, and we are even less aware of the impact which computing and the computer will eventually have upon mathematics and pedagogy. Today our mathematical instruction is barely beyond the pencil-and-paper and chalk-and-blackboard stage; relatively few mathematicians have had experience in mathematical modeling and in effective use of the computer.

Although it seems imperative today to re-examine the content of our courses and to give our students some training in the processes

by which mathematics is and can be applied, it is certainly beyond our experience at the moment to do so extensively at an elementary level. Modeling itself might best be introduced as an integral part of courses designed to teach a certain body of mathematics, but initially it might be better and easier to gain experience by experimenting with a separate course in mathematical-computer modeling at a post-calculus level. This could be a joint experimental undertaking by a number of faculty members and a few students; it should consist of the study and investigation in depth of a small number of carefully selected problems.

In selecting a problem one should take the following things into account:

(1) The problem should be easily stated. Without requiring extensive specialized knowledge or background, it should be possible to distinguish enough of the essential features in order to begin to construct some mathematical models, however crude.

(2) The problem should have mathematical content--the simpler the better at this level--which illustrates how mathematics is needed (i) to provide insight, (ii) to test the model (e.g., against a simple special case where the solution is obvious or easy), (iii) to develop a theory of the essential features of the model, and (iv) to indicate computational procedures.

(3) The problem should in some essential way require use of the computer (i) to provide insight through computer experimentation with the model or problem, (ii) to provide approximate answers and practical solutions, and (iii) to test the model and the solutions.

This does not imply that it is impossible to learn a great deal about modeling with pencil and paper, but a basic objective here is to go beyond this stage and to learn something about the uses and misuses of the computer and mathematical theory. More time needs to be spent in thinking about what goes into and what comes out of the computer than about the computation itself.

It is within the rules of the game to use mathematical or scientific results without proof, although where proofs are easily accessible and instructive they could be included. It would also be good pedagogy to consider models which are known to be poor, impractical theories and solutions, and poor numerical methods.

A SAMPLE PROBLEM

An excellent example is suggested by the work of Harold W. Kuhn. See his papers listed in the references at the end of this section; see also Courant and Robbins.

Fermat-Steiner-Weber Problem

Given n distinct points $p^1 = (x_1,y_1)$, $p^2 = (x_2,y_2)$, ..., $p^n = (x_n,y_n)$ in the plane and n positive numbers $w_1, w_2, \ldots, w_n$, find points which minimize

$$F(p) = \sum_{i=1}^{n} w_i |p - p^i| ,$$

where $p = (x,y)$ and $|p| = (x^2 + y^2)^{\frac{1}{2}}$ (thus $|p - p^i|$ is the Euclidean distance between p and p^i). This problem, posed by Fermat in the early 17th century with $n = 3$ and $w_1 = w_2 = w_3 = 1$, has had a long history and has been studied recently with renewed interest because of applications to spatial economics (optimal location of a factory, a shopping center, a hospital, a communications center, etc.).

Omitting the trivial case when the n points are collinear, we can show without difficulty that F is strictly convex, has a unique minimum which is in the convex hull of $p^1, p^2, \ldots, p^n$, and that the vanishing of a gradient (suitably defined at the vertices p^j) is a necessary and sufficient condition for a minimum.

The history and theory is interesting and provides a necessary background to the problem of finding approximate solutions numerically. The computational difficulties are nontrivial.

The following algorithm has been independently proposed at least three times:

Let

$$q^n = T(q^{n-1}),$$

where

$$T(q) = q + h(q)\, \nabla F(q),$$

$$h(q) = \left(\sum_{k=1}^{n} w_k \, |q - p^k|^{-1} \right)^{-1}$$

with $T(p^k) = p^k$ at the vertices [$h(q)$ is the harmonic mean of the distances to the vertices].

It can be shown that:

If q minimizes F, then it is a fixed point of T. If q

is a fixed point of T that is not a vertex, then q minimizes F. Either (1) $T^n(q)$ converges to a fixed point or (2) $T^j(q) = p^k$ for some j and some k.

Kuhn gives an algorithm which controls the step-size $h(q)\ \nabla F(q)$ for which he conjectures that $F(q^{n+1}) \leq F(q^n)$. This would imply convergence. Calculations involving $n = 3$ to $n = 24$ give close approximations after seven iterations.

Outline for the Study of this Problem

1. Nonmathematical statement and discussion of an economic problem involving the optimal location of a plant, shopping center, etc.

2. Mathematical statement of the problem. Locate in the plane a point that minimizes the weighted sum of its distances to n given points in the plane.

3. History of the problem. Solution of simple cases. Simplest case (3 points, equal weights) considered by Fermat (c. 1635) in an essay on maximum and minimum problems. The more general problem with weights w_1, w_2, w_3 appears in an early book on "fluxions" by Simpson, one of the first textbooks on calculus.

4. Some mathematical theory.
   a. Existence-uniqueness.
   b. Necessary and sufficient conditions.
   c. Dual problem.

5. Computational methods. Use of the computer.
   a. As a problem in mathematical programming.
   b. A proposed algorithm and its motivation. Iterations, convergence, and fixed points.
   c. Computation of some examples.
   d. The conjecture $F(q^{n+1}) \leq F(q^n)$. Special cases in which it can be verified.
   e. Computer tests of the conjecture.

6. A specific application. Study the problem of a good location for a large regional high school in the community.

7. Generalizations and unanswered mathematical and practical questions (research problems).

Desirable Features Illustrated by the Example

1. It is simple to describe, easily understood, explicit, interesting, and significant.

2. It has deep roots within the history of mathematics. Special cases of this problem appear as exercises in the earliest texts on "fluxions." It can be considered today in the light of new ideas, new mathematics, and computational procedures related to modern digital computers.

3. It serves to review and illustrate mathematics to which the student has been exposed: max-min, Lagrange multipliers (not required if the dual problem is omitted), simple linear algebra (analytic geometry), convergence.

4. It requires introduction at an elementary level of some new mathematics and new ideas important in mathematics and applications: convexity, duality, iteration (successive approximations), fixed points, and mathematical programming.

5. It provides an opportunity to develop a small body of theory.

6. It raises questions of computation, significant examples of which require the computer.

7. It raises a conjecture which can be proved in special cases and can be tested on the computer in more general cases.

8. It reaches the frontiers of research (generalizations to nonlinear costs, noneuclidean distance, etc., which are significant for applications).

A RECOMMENDATION

The development of individual topics, problems, exercises, etc., needed for a course of this type will require considerable work and imagination. This might be accomplished through isolated projects for independent group study with selected students, directed by an applications- and computer-oriented mathematician and a colleague representing the area of application.

Such experimental courses would be the testing ground for the development of instructional model building and are encouraged by the Panel. In the long run we believe that such model building should come in directly as a vehicle for teaching mathematics and its applications (for an example see the book by Grenander and the book by Freiberger and Grenander).

REFERENCES

Courant, Richard and Robbins, Herbert. What is Mathematics? New York, Oxford University Press, Inc., 1941, p. 354 ff.

Freiberger, Walter F. and Grenander, Ulf. A Short Course in Computational Probability and Statistics. New York, Springer-Verlag New York, Inc., 1971.

Grenander, Ulf. Computational Probability No. 10: Computational Probability and Statistics. Providence, Rhode Island, Brown University, 1971.

Kuhn, Harold W. "Locational problems and mathematical programming." Mathematical Optimization in Economics. Centro Internazionale Matematico Estivo, III Ciclo, L'Aquile, 29 Agosto-7 Settembre 1965. Roma, Edizioni Cremonese, 1966.

Kuhn, Harold W. "On a pair of dual nonlinear programs." Methods of Nonlinear Programming. Amsterdam, North-Holland Publishing Company, 1967, pp. 39-54.

Kuhn, Harold W. and Kuenne, Robert E. "An efficient algorithm for the numerical solution of the generalized Weber problem in spatial economics." Journal of Regional Science, 4 (1962), pp. 21-33.

SUPPLEMENTARY REFERENCES

The following books are meant to illustrate some areas and sources of ideas for modeling. Do not expect to find completely worked out instructional material.

Arrow, Kenneth J., ed. Selected Readings in Economic Theory from Econometrica. Cambridge, Massachusetts, MIT Press, 1971.

Athans, Michael A. and Falb, Peter. Optimal Controls. New York, McGraw-Hill Book Company, 1966.

Bellman, Richard. Introduction to the Mathematical Theory of Control Processes, vol. I. (Linear Equations and Quadratic Criteria). New York, Academic Press, Inc., 1967.

Canon, M.; Cullum, C.; Polak, E. Theory of Optimal Control and Mathematical Programming. New York, McGraw-Hill Book Company, 1970.

Computers in Undergraduate Science Education. Conference Proceedings, Chicago, Illinois, August 17-21, 1970. College Park, Maryland, Commission on College Physics, 1971.

Forester, Jay W. Urban Dynamics. Cambridge, Massachusetts, MIT Press, 1969.

Friedrich, O. "Population explosion: Is man really doomed?" Time, September 13, 1971, pp. 58-59.

Gerstenhaber, Murray, et al. AMS Lectures on Mathematics in the Life Sciences: Some Mathematical Problems in Biology, vol. I. Providence, Rhode Island, American Mathematical Society, 1968.

Hooper, John W. and Nerlove, Marc, eds. Selected Readings in Econometrics from Econometrica. Cambridge, Massachusetts, MIT Press, 1970.

IEEE Conference on Decision and Control. Institute of Electrical and Electronic Engineers, 345 East 47th Street, New York, New York 10017, 1971.

Joint Automatic Control Conference of the American Automatic Control Council. Institute of Electrical and Electronic Engineers, 345 East 47th Street, New York, New York 10017, 1971.

Kirk, Donald E. Optimal Control Theory, An Introduction. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1970.

Koenig, H. E., et al. Analysis of Discrete Physical Systems. New York, McGraw-Hill Book Company, 1967.

Ledley, Robert S. Use of Computers in Biology and Medicine. New York, McGraw-Hill Book Company, 1965.

Meyerson, Martin and Banfield, Edward C. Politics, Planning, and the Public Interest. New York, Free Press, 1955.

Noble, Ben. Applications of Undergraduate Mathematics in Engineering. New York, The Macmillan Company, 1967.

Pielou, E. C. An Introduction to Mathematical Ecology. New York, John Wiley and Sons, Inc., 1969.

Polak, E. Computational Methods in Optimization. New York, Academic Press, Inc., 1971.

Shapiro, George, et al. Prospects for Simulation and Simulators of Dynamic Systems. New York, Spartan Books, Inc., 1967.

Solow, Robert M. "The economist's approach to pollution and its control." Science, 173 (1971), pp. 498-503.

Steinitz, Carl and Rogers, Peter. Systems Analysis Model of Urbanization and Change. Cambridge, Massachusetts, MIT Press, 1970.

Sternberg, S. "Stochastic Learning Theory." Handbook of Mathematical Psychology, vol. II. New York, John Wiley and Sons, Inc., 1963.

Wagner, Harvey M. Principles of Operations Research with Applications to Managerial Decisions. Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1969.

Wodwell, G.; Craig, P.; Johnson, H. A. "DDT in the biosphere: Where will it go?" Science, 174 (1971), pp. 1101-1107.

Zadeh, Lotfi and Polak, Elijah. System Theory. New York, McGraw-Hill Book Company, 1969.

4.4 Thoughts on a "Postponed" Calculus Course with Emphasis on Numerical Methods

In recent years many questions have been raised about the special role played by the basic calculus sequence as the first set of courses in traditional college mathematics curricula. There are many arguments for beginning with the calculus, but with the growth of computer science and the need for more mathematics in the behavioral and social sciences there are more and more arguments for postponing the calculus courses.

For those students who do not need to use the calculus in other courses until the junior or senior year, a drastically revised one-year calculus course which makes heavy use of computing and algorithmic ideas may be suitable. This course would have the Discrete Mathematics course MC-DM and a thorough knowledge of programming as prerequisites and would not be taken until the sophomore or junior year. A constructive approach to the basic concepts of the calculus would be used throughout the course and heavy emphasis would be placed on both numerical and nonnumerical algorithms. The course would contain some elementary numerical analysis, attention being paid to error analysis and degrees of approximation.

By necessity, some of the traditional topics of the calculus will have to be slighted, but the knowledge that the students will gain in being able to handle fairly complex real-world problems would certainly offset this.

The following outline should be considered as a first tentative suggestion. Given the novelty of the approach, there are very few experiences which might have been used as a guide. The material is ample for a one-year course, but no attempt is made to indicate the pace. The increased mathematical maturity of the students should make possible a faster pace than that in the usual calculus course. It should be kept in mind throughout the course that the topics are to be treated with heavy emphasis on numerical orientation.

A TENTATIVE OUTLINE

1. Numbers. A brief review of (intuitive) number concepts. Distribution of floating-point numbers on the line. Arithmetic problems with floating-point numbers. Roundoff errors. Ordering, inequalities, distances, and absolute value. All of this should be computationally oriented.

2. Sequences. Computational example of approximating the square root. Squeeze concept. Other related examples of limits. Need for irrational numbers to "fill" the number line; completeness concept. Definition of limit. Basic limit theorems (prove only a few). Squeeze theorem. Importance of error estimates. Slow and fast convergence illustrated by various examples.

3. Functions. Review of function concept (functions as mappings). Functions defined by algorithms, e.g., Horner's scheme, etc. Graphs and basic curve sketching. Arithmetic combination of functions. Geometric discussion of "near" functions and simple computational examples of approximations. Composition of functions, inverses. Monotonicity. Zeros, bisection algorithm. Uniform continuity; Intermediate Value Theorem for uniformly continuous functions using bisection algorithm.

4. Interpolation. Polynomial interpolation, undetermined coefficients, Lagrange formula, application to the solution of equations.

5. Derivatives. Limits, basic limit theorems with reference to the sequential case. Motivation and definition of the derivative. The cases x^k (small k) and $1/x$. Concept of higher derivatives. Continuity. Basic differentiation theorems. Derivative of polynomials, Horner's algorithm again. Derivative of rational functions. Linearization. Newton's method. Monotone convergence of Newton's method. Derivatives of monotone, convex, and concave functions. Chain rule. Implicit functions, inverse functions, application to $x^{1/n}$.

6. Area. Intuitive discussion of properties of area. Area of regions under monotone functions by approximations with sums of rectangles. Extension to nonmonotone functions, application to x^k, $k = 0, 1, 2, 3$.

7. Integral. Riemann sums, existence for uniformly continuous functions, basic properties. The Fundamental Theorem of Calculus. Application to the calculation of definite integrals. Substitution and integration by parts.

8. Quadrature. Review of rectangular approximation, trapezoidal rule and Simpson's rule. Integration by the Lagrange formula. Algorithmic treatment of partial fractions.

9. Differential equations. Direction fields, concept of solving first-order initial value problems. Separable case. Euler's method. Differential equations of radioactive decay, first-order logarithms. Second-order linear equations, superposition principle, harmonic motion, trigonometric functions.

10. Taylor's theorem. Mean Value Theorem, Taylor's theorem. Lagrange and integral remainder, application to error of interpolation, quadrature, l'Hôpital's rule, critical points, simple numerical methods for critical points.

11. Numerical solution of differential equations. Euler's method reviewed, trapezoidal rule, local discretization error, modified Euler's methods, Taylor's polynomial methods, idea of Runge-Kutta and multi-step methods. Brief geometric discussion of stability problems.

REFERENCES

There is no single textbook which covers the material proposed here, but parts of the following three texts may be used.

Flanders, Harley; Korfhage, Robert R.; Price, Justin J. Calculus. New York, Academic Press, Inc., 1970.

Henriksen, M. and Lees, M. Single-Variable Calculus. New York, Worth Publishers, Inc., 1970.

Stenberg, Warren, et al. Calculus, A Computer Oriented Presentation, Parts 1 and 2. CRICISAM, Florida State University, 1970.

For the numerical analysis portions, parts of various standard books on numerical methods can be used, especially for problems and applications.

4.5 Algebra Courses Influenced by Computing

At the present time it is not clear how the standard undergraduate introduction to algebra (e.g., Mathematics 6M in the GCMC Commentary) should be modified to reflect the growing influence of computers. Some knowledge of algebra is essential for an understanding of areas such as algebraic algorithms and symbol manipulation which have a strong algebraic flavor. Nonetheless, there is no consensus as to how the usual introduction to abstract algebra should be modified. In the discussion below we present brief outlines of three possible modifications, along with some sources of further information.

1. At Harvard University the Department of Applied Mathematics has taught a one-year course based on Birkhoff and Bartee, Modern Applied Algebra (New York, McGraw-Hill Book Company, 1970). Topics are selected from among the following:

> Sets and functions, relations, graphs. Finite state machines, programming languages. Monoids, groups, lattices, Boolean algebras, rings, polynomials, finite fields. Optimization and computer design, binary group codes, polynomial codes, recurrent sequences, computability.

For further details about the course, the book by Birkhoff and Bartee should be consulted.

2. Professor John Lipson of the University of Toronto has taught a modification of the one-year algebra course to advanced students in computer science for the past three years. Lecture notes for this course are expected to be available to interested parties sometime in 1973.

The principal topic in the second half of this course is a study of algebraic algorithms which incorporates recent work not readily available in the textbook literature. The following topics are considered:

> Sets, relations, functions. Examples of algebraic systems. Universal algebra. Lattices, Boolean algebra, groups, rings, finite fields. Interpolation theory, algebraic algorithms.

In addition to the usual textbooks in algebra, the following sources are used:

Berztiss, A. T. Data Structures. New York, Academic Press, Inc., 1971.

Birkhoff, Garrett. Lattice Theory, 3rd ed. Providence, Rhode Island, American Mathematical Society, 1966.

Birkhoff, Garrett and Bartee, Thomas C. Modern Applied Algebra. New York, McGraw-Hill Book Company, 1970.

Knuth, Donald E. The Art of Computer Programming. Reading, Massachusetts, Addison-Wesley Publishing Company. Vol. I, 1968; Vol. II, 1969.

3. A one-semester modification of the course described has been taught in the Department of Electrical Engineering at the Massachusetts Institute of Technology and in the Division of Applied Mathematics at Brown University. The following topics are covered:

> Sets, relations, functions, morphisms, diagram graphs and applications. Monoids, groups, lattices. Finite state machines, semantics of flow diagrams, programming languages. Rings, fields, polynomials, extension fields, finite fields.

5. Implementation

5.1 Computing Facilities

See Section 3.2 of Recommendations for an Undergraduate Program in Computational Mathematics, page 547.

5.2 Programming Requirements

The principal objective of any of the courses described in this report is to describe a mathematical subject area and applications related to it. Thus, the teaching of programming should not, by itself, be a purpose of any of these courses. Ideally, a student entering any of the lower-division courses except MC-0 should be required to have at least a beginning knowledge of one of the standard algorithmic languages implemented at his institution, as well as the ability to develop flowcharts and basic programs from a general description of a process. For the upper-division courses a more thorough familiarity with such a language and more programming expertise is required.

At present, few students entering the lower-division courses will have the corresponding programming background, although the expanding use of computers in high schools may change this picture in the future. Meanwhile, there are several alternatives that can be adopted.

If a student's schedule permits, one solution would be for him to take a one-semester introduction to computing, such as the course C1 described in the CUPM report Recommendations for an Undergraduate Program in Computational Mathematics. If this approach leads to delays in the mathematical progress, a possible alternative might be

to let him take the computing course and his first mathematics course concurrently. In that case any one of the courses in Section 3 could be modified and taught in such a way that programming is not absolutely essential, although the results of the computations and the problems raised by computations would, of course, be used in the course.

Another alternative not involving a separate computing course is to teach a minimum amount of programming in supplementary lectures to those who need it during the first few weeks of the freshman courses. The time required for this depends considerably on the computing facilities available and on the language used; here computer use in a conversational mode is often particularly helpful. It is essential that the students be given ample opportunity to write and run programs of their own and to operate the necessary equipment, such as terminals or key punches. Moreover, it is important that consultants be available who can help them over their difficulties without overwhelming them with technical details. In courses where additional credit is given for the computational work, the supplementary programming lectures would, of course, take up the first few of the laboratory sessions held throughout the semester.

Which of these alternatives is the most feasible in a given situation depends not only upon the intended use of the computer in the course but also upon the nature of the available computing facilities.

As mentioned before, a few lectures in programming are not sufficient preparation for the more advanced courses. A consistent programming experience in the lower courses may, in general, enable a student to read an introductory computer science text on his own and to round out his computer knowledge in this way. Otherwise, a first computing course such as the course C1 cited earlier is certainly a natural prerequisite for the upper-division courses.

5.3 Changes in Instructional Techniques

In connection with the general topic of this report it is appropriate to review the state of teaching techniques in light of requirements for incorporating computers into the curriculum and to develop new teaching methods for bringing computational results and numerical algorithms into the classroom. The principal objective is to foster the "laboratory" atmosphere in class and to make each student feel that he is actively engaged in learning through problem solving, experimentation, and discovery.

It is important to bring the computational results into the classroom. Although thoughtful students will learn well from programming projects assigned as homework, the hurried or less thoughtful students see these assignments as chores to be done as quickly as possible. Sometimes a student will turn in a program with an error

in it so gross as to make his answers meaningless. He will not have learned anything from the activity unless the instructor is able to review the work in class and exhibit the results which the problem was intended to elucidate.

The college mathematics teacher has always been at a real disadvantage when asked to make his lectures with chalk and blackboard as exciting and interesting as those of, say, his colleagues in physics who have carefully orchestrated, and often dramatic, experiments to perform in class. More than ever, though, we find chalk and blackboard inadequate for the presentation of the new material being proposed in this report; a teacher filling a board with computer results to six significant digits is likely to deter even the most energetic student! We hope that authors and publishers will address themselves to this problem and begin to develop new teaching materials for the mathematics teacher. Three possibilities are mentioned below, in order of increasing cost and complexity.

The first and most accessible teaching assist might come from sets of transparencies to be used with an overhead projector. Graphs of functions of one and two dimensions, successively "blown-up" portions of them, and computational results can all be presented. Carefully prepared overlays can give graphical results a dynamic sense. We are all familiar with the power and appeal of really good, professionally executed illustrations in textbooks. A library of transparencies of equal excellence with which a teacher could illustrate his lecture would go far toward livening up the classroom. The teacher interested in developing visual material should seek help from a media specialist.[1]

The second possibility to be considered is that of videotaped or filmed presentations. Here the dynamic nature of the algorithms can be well conveyed. For illustration, let us consider the concept of the definite integral. If the limit definition is phrased in an algorithmic form, the student will comprehend it best if he sees the approximating rectangles sketched, their areas added in one at a time, and the whole process repeated for a finer partition. When the partition is refined, he sees the effect of taking a larger number of smaller contributions to the integral. It is very difficult to draw accurately enough and fast enough on a blackboard to give students this sense of dynamism. Also, when animation is under consideration, it is natural to try to incorporate computer-produced graphics in these presentations.[2]

1. Some information might also be obtained from the Association for Educational Communications and Technology, 1201 16th Street, N.W., Washington, D. C. 20036.

2. Advice may be obtained from Educational Development Center, 55 Chapel Street, Newton, Massachusetts 02160.

An independent reason for developing recorded presentations is that television cassette technology is reaching a stage which will allow a student to view a presentation independently, making individualized instruction a reality. A "library" of cassettes will make it possible for him to spend as much time as necessary on precisely the material that is appropriate for him. Courses could become modular in nature and it would no longer be necessary for all students to proceed in lock-step through the material. We note that freshman classes are becoming increasingly heterogeneous, both with respect to the students' capabilities and to the quality and quantity of their high school mathematics preparation. As "learning centers" with carrels containing TV screens and other audio-visual devices become increasingly common, the mathematical community should be concerned with their potential impact and usefulness.

We encourage authors who wish to prepare materials utilizing these new media to seek professional help from audio-visual specialists. Television and film offer new opportunities for innovative teaching. Simply to televise or to film traditional lectures would fail to take full advantage of the possibilities afforded by these media.

The third and most sophisticated and desirable technological solution is to have an on-line terminal connected to a reliable computer available at all times in the classroom. Devices are available which tap the input to a cathode ray tube display device and put the same image on one (or more) television monitors so that a large class can "participate" in the interaction.[3] If an on-line computer is used, a great deal of preliminary work is required on the part of the teacher. Numerical experiments must be chosen with great care, lest roundoff errors, the peculiarities of the computer operating system, etc., produce unanticipated results. Thus "inverting" a nearly singular matrix or "summing" an alternating series with terms alike to 6 digits using 5-digit arithmetic would obscure rather than illuminate, and could carry the teacher far deeper into the theory of computation than he ever intended to go. These problems are particularly likely to arise if a mini-computer with a small word length is used with only single precision arithmetic.

3. For an overview of these technological developments, we recommend Ronald Blum, ed., Computers in Undergraduate Science Education Conference Proceedings, Commission on College Physics, College Park, Maryland, 1971 (available from American Institute of Physics, 335 East 45th Street, New York, New York 10017). See also Proceedings of a Conference on Computers in the Undergraduate Curricula, 1970 (available from the University of Iowa Computer Center, Iowa City, Iowa 52240), Proceedings of the Second Annual Conference on Computers in the Undergraduate Curricula, 1971 (available from The New England Press, Box 979, Hanover, New Hampshire 03755), and Proceedings of the 1972 Conference on Computers in the Undergraduate Curricula, 1972 (available from Southern Regional Education Board, 130 Sixth Street, N.W., Atlanta, Georgia 30313).