An Undergraduate Program in Computer Science—Preliminary Recommendations

A Report from the
ACM Curriculum Committee on Computer Science

The Curriculum Committee on Computer Science (C³S) of the Association for Computing Machinery has been considering curriculum problems for approximately three years. During the early part of this period, a number of informal sessions were held with computer people at various national meetings. In the latter part of this three-year period, the Committee has been formally organized and has made a definite effort to arrive at concrete suggestions for a curriculum.

This report is a result of these latter efforts. During the summer of 1964, a one-week session was held by the Committee at The Homestead in Poughkeepsie, New York, as guests of the International Business Machines Corporation. While a series of earlier sessions had been held on curriculum definition, our first formal writing was done at the Poughkeepsie meeting. Since that time, several complete meetings or partial meetings have been held to complete the recommendations.

The Committee does not consider these suggestions to be final, but only as a point of departure. We hope that these recommendations will stimulate comment and suggestions for their further improvement.

The following curriculum has already had the benefit of critical review by a large number of computer scientists. The number of these people is far too large to permit a listing.

Our formal working committee consisted of:

S. D. Conte, Purdue University
John W. Hamblen,* Southern Illinois University
Thomas A. Keenan, University of Rochester
William B. Kehl, University of Pittsburgh
Silvio O. Navarro,† University of Kentucky
Werner C. Rheinboldt, University of Maryland
Earl J. Schweppe, University of Maryland
David M. Young, Jr., University of Texas
William F. Atchison, Georgia Institute of Technology (Chairman)

Special mention should be made of Dr. Elliott I. Organick (University of Houston), Dr. C. C. Gotlieb (University of Toronto), Dr. Thomas E. Hull (University of Toronto), and Dr. George E. Forsythe (Stanford University) for their help. Each of them attended part of our formal sessions, the latter having been an actual member of our Committee. Our Committee is also indebted to George Heller for his assistance and encouragement while he was Chairman of the Education Committee and our group was a subcommittee of his Education Committee.

INTRODUCTION

The advent of the computing machine and related automata has brought about a rapid growth in new areas of research and the need for considerable modification in the educational offerings of our colleges and universities. Although much change has been accomplished within existing programs, such as mathematics and electrical engineering, there is a sizeable area of work which does not naturally fit into any existing field. Thus, it is now generally recognized that this area, most often called Computer Science [1],[1] has become a distinct field of study. This development is reported by the Committee on Uses of Computers of the National Academy of Sciences [2], by several individual authors [3, 4, 5, 6, 8], and in a report of the Committee on the Undergraduate Program in Mathematics (CUPM) [7] which states:

> Out of the solution of such problems as these has emerged a field of study called Computer Science, embracing such topics as numerical analysis, theory of programming, theory of automata, switching theory, etc.

Reprints are available at nominal costs from ACM Headquarters (211 E. 43 St., New York, N. Y. 10017).

* Present affiliation: Southern Regional Education Board, Atlanta, Georgia.

† Dr. Navarro, of the University of Kentucky, has just recently joined the Committee.

[1] These numbers refer to a list of general references which appear immediately before the section on Computer Science Courses.

There are many schools already giving undergraduate degrees in Computer Science or who are about to give such degrees. It is the hope of the Curriculum Committee on Computer Science of ACM that our work and the recommendations included in this report will give some guidance to such schools and will serve as a coordinating force for the various diverse efforts.

We recognize that many needs are to be met in computer-related education; instruction in the use of computers as a tool is increasingly sought by students in the natural sciences, literature, medicine, management, etc. However, the background of the students, the language appropriate to the subject, the pertinent exercises and examples, all differ, depending on the students' primary field. The Committee has chosen to direct its attention *first* to the student whose primary interest is in computer science. The Committee solicits comment and criticism of the present report with the view of both strengthening the current recommendations and illuminating the relation between computer science and other fields of study.

The report of CUPM entitled "Recommendations on the Undergraduate Mathematics Program for Work in Computing" is very well done. Members of C^{3}S, some of whom had consulted with CUPM, were greatly pleased with the perception shown in this document and with the fact that the plan suggested by CUPM to some degree paralleled their own thinking. That report, however, was for the mathematics major and we feel that additional work is needed for the computer science major, especially in view of the fact that so many schools are moving toward such a major [9].

Objectives

The curriculum is intended to be flexible enough that students receiving the baccalaureate can follow one of several paths. Among these are:

1. Undertake graduate work in computer science. It is hoped that many of the best students will follow this path since the future leadership of the computer science community depends on it.
2. Contribute in the rapidly growing profession of systems programming.
3. Work on applications programming. It is felt that an undergraduate program would provide a stronger background in numerical analysis, logic, statistics, and formal languages than that available to many of those now engaged in applications programming.
4. Undertake graduate work in a field other than computer science. It is expected that students following this path would help introduce the methodology and discipline of computer science into other fields.

In order that graduates of the program be prepared to follow the alternative plans listed above, the curriculum we are recommending contains a small group of required courses that every computer science major should master and a selection of courses to be elected (with guidance) in the junior and senior years. All the courses to be described in this report are based on courses in existence, or in some instances on material known to be essential to a well trained computer scientist but not yet fully developed into complete courses. This latter material is to be the object of further work of the Committee. An immediate objective is to organize known material into a rational academic curriculum.

Computer Science as a Discipline

There are widespread misconceptions of the purpose of computer science despite the general acknowledgement that it is a distinctive subject. Computer science is not simply concerned with the design of computing devices—nor is it just the art of numerical calculation, as important as these topics are. Computer science encompasses many specialized areas, all developing with extreme rapidity. It is natural that one should associate some small part, to which he has been exposed, with the subject as a whole. Even among those specializing in computer science, few (if any) are able to keep pace with the flood of innovation throughout the field.

Computer science is concerned with *information* in much the same sense that physics is concerned with energy; it is devoted to the *representation*, *storage*, *manipulation* and *presentation* of information in an environment permitting automatic information systems. As physics uses energy transforming devices, computer science uses information transforming devices. Some forms of information have been more thoroughly studied and are better understood than others; nevertheless, all forms of information—numeric, alphabetic, pictorial, verbal, tactile, olefactory, results of experimental measurement, etc.—are of interest in computer science.

Mathematics, too, is concerned with information and its structure and this tends to confuse those not well versed in both mathematics and computer science. The mathematician is interested in discovering the syntactic relation between elements based on a set of axioms which may have no physical reality. The computer scientist is interested in discovering the pragmatic means by which information can be transformed to model and analyze the information transformations in the real world. The pragmatic aspect of this interest leads to inquiry into effective ways to represent information, effective algorithms to transform information, effective languages with which to express algorithms, effective means to monitor the process and to display the transformed information, and effective ways to accomplish these at reasonable cost.

The fact that computer science is now a distinct academic discipline is demonstrated by the rapidly increasing number of colleges and universities [9] which have established departments of computer science. At the time of this writing, in the United States, doctorates can be sought at more than 15 universities, master's degrees at more than 30, baccalaureates in at least 17 colleges, and a sizeable number of other colleges and universities are presently

planning or considering departments of computer science [10].

Description of the Report

Catalog-type course descriptions together with prerequisites are given in the section entitled "Computer Science Courses." In some cases we have had to decide whether a course should be listed as a computer science course, and included in that section or be assigned to the role of a supporting course from another discipline. This choice has been made based on the relevancy to the entire computer science program. Certain courses can be distinguished from similar courses in other disciplines by the prerequisites and experience required of the student, and the relative emphasis and approach to the topics. For example, a course in logic obviously can be taught without any reference to computers.

A description of the curriculum, the elective options and the supporting courses, is given in the section entitled "Computer Science Curricula and Supporting Courses." In that section a distinction is made between the required courses, the electives that are not required but are important enough to recommend strongly, and electives of a more specialized nature.

More complete specifications of the computer science courses are given in the subsection "Descriptions of Computer Science Curriculum Courses—Outlines and References." Each of these descriptions is a detailed list of topics that could be dealt with in a one-semester course. We have not tried to say how much time should be spent on each topic. A list of references is given with each course description. These references are not necessarily intended as recommended texts for the course. Adequate texts are not yet available for some of the courses.

Implementation of this Curriculum

The computer science curriculum is presented here for a college using the semester system where each semester "course" normally carries three hours of credit. Because of the diversity with which college curricula are organized in the United States, many colleges will need to interpret this curriculum for their organization. Other colleges will, of course, find that they are not able to offer all the courses (especially among the electives) or, perhaps, that other electives are better suited to the interests of their faculty and the needs of their students.

In applying these recommendations to any given college, consideration should clearly be given to the possibility that certain courses may already be available or may be made available by modification of existing courses. Double listing of courses is a technique that has sometimes been used with apparent success.

We cannot be specific with regard to faculty organization for this curriculum simply because of the diversity of the American college structure. In some schools an independent department of computer science may be appropriate; in others such a department of computer science may not. We concur with CUPM that those responsible for the computer science curriculum should be closely linked to the computer scientists engaged in providing computational facilities. Although we further agree that most of the computer science courses described in this report will not be taught within existing mathematics departments, there is considerable benefit to be gained from mutual cooperation. In any case either new departments will need to be established, or existing mathematics (or other) departments will need to recognize the participation of computer scientists in their faculty development plan.

The Committee believes that several of the courses described in these recommendations will be of value to students whose prime interest is in another area of study. Students in engineering or any of the sciences will find Introduction to Algorithmic Processes (Course Number 1) a valuable introduction to the use of computers which provides an insight into computer science. Students interested in following the CUPM recommendations while majoring in mathematics will find ample material in our recommendations. Engineering students may find Courses 6 and 10 particularly valuable. Numerical Calculus and Numerical Analysis (Courses 3, 7, and 8) are of increasing value to students of many fields.

We have not investigated the question of whether some modification of standard courses in mathematics, physics, etc., would be desirable from the viewpoint of a computer science major. Some intuitively feel that this may be the case but admit that the question needs careful consideration.

Future Work of the Committee

The work of C^3S is in its early stages. Readers of this report are encouraged to communicate their ideas, experiences, and criticisms to the Committee. Not only will the present recommendations need periodic review but many suggestions concerning further computer-oriented curricula are sought. Study of the computer science needs of students in the nonphysical sciences is appropriate. It has been suggested that the educational needs of those who will plan and design the computing and communication equipment of the future should be given special consideration. More specific thought will be given to the education of those wishing to prepare themselves for work in information retrieval, management science, the life sciences, and the behavioral sciences. Finally, the present recommendations do not deal with graduate computer science. A committee to study graduate curriculum is in the process of formation.

General References

1. Atchison, W. F., and Hamblen, J. W. Status of computer sciences curricula in colleges and universities. *Comm. ACM* 7 (Apr. 1964), 225–227.
2. National Academy of Sciences—National Research Council, Committee on Computer Uses. Computer needs in American colleges and universities (a report). Publ. Dept., NAS, Washington, D. C., 1964.

3. GORN, S. The computer and information sciences: a new basic discipline. *SIAM Rev. 5* (Apr. 1963), 150–155.
4. FEIN, L. The role of the university in computers, data processing and related fields. *Comm. ACM 2* (Sept. 1959), 7–14.
5. KEENAN, T. Computers and education. *Comm. ACM 7* (Apr. 1964), 205–209.
6. FORSYTHE, G. E. An undergraduate curriculum in numerical analysis. *Comm. ACM 7* (Apr. 1964), 214–215.
7. MATHEMATICAL ASSOCIATION OF AMERICA, COMMITTEE ON THE UNDERGRADUATE PROGRAM IN MATHEMATICS. Recommendations on the undergraduate mathematics program for work in computing. The Committee, P. O. 1024, Berkeley, Calif., May 1964.
8. FORSYTHE, G. E. University education in computer science. Paper presented at the Fall Joint Comput. Conf., San Francisco, Calif., Nov. 1964.
9. SIXTH SURVEY OF UNIVERSITY COMPUTING FACILITIES. Prepared under the direction of Dr. T. A. Keenan, U. of Rochester, July 1963.
10. PANEL ON COMPUTER SCIENCE CURRICULUM. Proc. ACM Nat. Conf., Aug. 1964. Papers by E. J. Schweppe, S. D. Conte, R. S. Varga.

COMPUTER SCIENCE COURSES

Table of Courses for Computer Science Majors

TABLE 1. PRELIMINARY RECOMMENDATIONS OF THE CURRICULUM COMMITTEE OF ACM FOR MAJORS IN COMPUTER SCIENCE

Courses / Recommendations	*Computer Science*				*Supporting*
	Basic Courses	*Theory Courses*	*Numerical Algorithms*	*Computer Models and Applications*	
Required	**1.** INTRODUCTION TO ALGORITHMIC PROCESSES* **2.** COMPUTER ORGANIZATION AND PROGRAMMING **4.** INFORMATION STRUCTURES	**5.** ALGORITHMIC LANGUAGES AND COMPILERS	**3.** NUMERICAL CALCULUS (or Course 7)		Beginning Analysis (12 cr.) Linear Algebra (3)
Highly Recommended Electives	**6.** LOGIC DESIGN AND SWITCHING THEORY **9.** COMPUTER AND PROGRAMMING SYSTEMS		**7.** NUMERICAL ANALYSIS I **8.** NUMERICAL ANALYSIS II		Algebraic Structures Statistical Methods Differential Equations Advanced Calculus Physics (6 cr.)
Other Electives	**10.** COMBINATORICS AND GRAPH THEORY	**13.** CONSTRUCTIVE LOGIC **14.** INTRODUCTION TO AUTOMATA THEORY **15.** FORMAL LANGUAGES		**11.** SYSTEMS SIMULATIONS **12.** MATHEMATICAL OPTIMIZATION TECHNIQUES **16.** HEURISTIC PROGRAMMING	Analog Computers Electronics Probability and Statistics Theory Linguistics Logic Philosophy and Philosophy of Science

* The specifications for the 16 Courses, indicated by boldface numerals, are given in this report.

Catalog Descriptions of Computer Science Courses and Prerequisites

REQUIRED COURSES:

1. Introduction to Algorithmic Processes (2-2-3)[2]
Prerequisite: Concurrent registration in Beginning Analysis
Concept and properties of an algorithm, language and notation for describing algorithms, analysis of computational problems and development of algorithms for their solution, application of a specific procedure-oriented language to solve simple numerical and non-numerical problems using a computer.

[2] The first number gives the number of lectures, the second the number of laboratory hours, and the third the number of hours of credit.

2. Computer Organization and Programming (2-2-3)
Prerequisite: Course 1 above
Logical basis of computer structure, machine representation of numbers and characters, flow of control, instruction codes, arithmetic and logical operations, indexing and indirect addressing, input-output, subroutines, linkages, macros, interpretive, and assembly systems, pushdown stacks, and recent advances in computer organization. Several computer projects to illustrate basic concepts will be incorporated.

3. Numerical Calculus (2-2-3)
Prerequisite: Course 1 above and Beginning Analysis 3
An introduction to numerical methods. Includes elementary discussion of errors, polynomial interpolation, quadrature, linear systems of equations, solution of nonlinear equations, and numerical solution of ordinary differential equations.

The algorithmic approach and the efficient use of the computer will be emphasized.

4. **Information Structures (2-2-3)**
   Prerequisite: Course 2
   Study of information representations and relationships between the form of representation and processing techniques. Transformations between storage media. Referencing of information as related to the structure of its representation and implications for the design of the referencing language.
5. **Algorithmic Languages and Compilers (3-0-3)**
   Prerequisite: Course 2 above and possibly Course 4
   Formal description of algorithmic languages, e.g., ALGOL, and the techniques used in their compilation. Study of syntax, semantics, ambiguities, procedures, replication, iteration and recursion in these languages. Syntactic decomposition and the theory of compilers which are syntax directed or recursively controlled.

ELECTIVE COURSES:

6. **Logic Design and Switching Theory (3-0-3) or (2-2-3)**
   Prerequisites: Courses 1, 2
   Symbolic logic and Boolean algebra for description and analysis of switching circuits; simplification of switching circuits; error detecting and correcting codes; storage elements defined logically; basic sequential circuits; digital systems design principles.
7. **Numerical Analysis I (3-0-3)**
   Prerequisites: Course 1 above and Advanced Calculus
   A thorough treatment of solutions of equations, interpolation and approximations, numerical differentiation and integration, numerical solution of initial value problems in the solutions of ordinary differential equations. Selected algorithms will be programmed for solution on computers.
8. **Numerical Analysis II (3-0-3)**
   Prerequisites: Course 1 above, Advanced Calculus and Linear Algebra
   The solution of linear systems by direct and iterative methods, matrix inversion, the evaluation of determinants, eigenvalues and eigenvectors of matrices. Application to boundary value problems in ordinary differential equations. Introduction to the numerical solution of partial differential equations. Selected algorithms will be programmed for solution on computers.
   NOTE: While other arrangements of the material in Courses 7 and 8 are possible, the arrangements suggested here allow the two courses to be taught independently. It may also be considered desirable to require Numerical Calculus as a prerequisite for these courses.
9. **Computer and Programming Systems (3-0-3)**
   Prerequisite: Course 4 above
   Input-output and storage systems, structures and transformations of data bases, assembly and executive systems.
10. **Combinatorics and Graph Theory (3-0-3)**
    Prerequisite: Course 1 and Beginning Analysis
    An introduction to set theory, graph theory, and combinatorial analysis. Includes such topics as set algebra, order relations, cardinality, indirected and directed graphs, elementary combinatorics, principle of inclusion and exclusion, recurrence relations, zero-one matrices, partitions, and Polya's Theorem.
11. **Systems Simulations (2-2-3)**
    Prerequisite: Course 1, possibly Course 4, and Probability and Statistics
    Computer simulation utilizing logical, numerical and Monte Carlo modeling to represent systems. The description of the status of systems by the use of sets of entities and the modification of this status by events. The generation, termination, and flow of entities possessing prescribed attributes through storage and processing facilities. Balancing systems, sharing facilities and using priorities to modify performance. Collection and evaluation of statistics on passage times, flow volumes, queue lengths, manpower and equipment utilization. Use of special simulation languages to simulate actual systems.
12. **Mathematical Optimization Techniques (3-0-3)**
    Prerequisite: Course 8
    Extremal properties of multivariate functions with and without constraints, convex sets and convex functions, linear programming, quadratic programming, dynamic programming.
13. **Constructive Logic (3-0-3)**
    Prerequisite: Course 1 and Beginning Analysis
    An introduction to logic and Boolean algebra. Set algebra, product sets, relations and functions, propositional calculus, algorithms and quantification theory.
14. **Introduction to Automata Theory (3-0-3)**
    Prerequisite: Course 13, and Algebraic Structures
    A discussion of various types of automata, such as finite, probabilistic, growing, and reproducing automata. Representation of automata by regular expressions, state graphs, logical nets, recursive functions, and Turing machines.
15. **Formal Languages (3-0-3)**
    Prerequisites: Course 5 and possibly Course 13
    This course is a study of certain languages and grammars which can be specified in precise mathematical terms. These formal languages are closely related to computer languages and can serve as primitive models of natural languages. Various types of formal languages are defined and their properties derived. The ideas have been applied in computer programs for syntactic analysis and machine compilation.
16. **Heuristic Programming (3-0-3)**
    Prerequisites: Courses 4 and 11
    Distinction between heuristic and algorithmic methods. Polya's and Hadamard's role in mathematical intuition. Justification of the need for heuristic approach. The objectives of work in Artificial Intelligence and in Simulation of Cognitive Behavior. Discussion of research projects using heuristic programming techniques. Brief description of related research areas, such as Self-Organizing Systems, Neural Cybernetics, Automata Theory, Machine Translation of Languages, Information Retrieval, Automatic Medical Diagnosis, Machine Composition of Music, Automatic Engineering Design, etc.

COMPUTER SCIENCE CURRICULA AND SUPPORTING COURSES

Requirements and Options for Computer Science Majors

It is not intended by the Committee that its recommendations be iron-clad, but instead that they form a realistic basis upon which a sound program in computer science can be structured. Certain deviations will be necessary, depending upon the particular institution, its faculty and its objectives. Since the Committee's recommendations are presented in terms of a semester basis, certain changes will be necessary by an institution which operates under a quarter system.

The courses displayed in Table 1 summarize the Committee's recommendations. The courses have been classified horizontally as Computer Science and Supporting and vertically as being Required, Electives and Highly Recommended Electives.

Within computer science we have identified four topic areas[3] under which we have chosen to group these courses.

This arrangement is primarily to clarify the continuity within the course program and to suggest a possible structure for organizing graduate curricula in computer science to extend this program. The higher numbered courses build cumulatively on the knowledge from the earlier courses in each topic. But these areas are obviously not mutually exclusive.

The combined total of the required courses and the highly recommended electives amount to approximately 54 semester credit hours. Several of the supporting courses will usually satisfy other general requirements which means that a student will have considerable flexibility in completing a program. A few of the many possible options are the following:

(a) *Numerical Analysis Option:* One semester of advanced calculus and Numerical Analysis I and II.

(b) *Statistics Option:* Two courses in statistical theory and two in statistical methods.

(c) *Electronics:* An additional two to three courses in electronics.

(d) *Physical Sciences:* Two courses in each of two physical sciences.

(e) *Mathematical Linguistics:* Courses as they are available.

Descriptions of Computer Science Curriculum Courses—Outlines and References

1. Introduction to Algorithmic Processes (2-2-3)

This is the basic course for students expecting to major in computer science. It is intended to:

(A) Introduce the student to the intuitive notion of an algorithm. Emphasize the power of algorithmic descriptions of problem solutions and the requirements for well-formed representations of algorithms.

(B) Introduce a procedure-oriented language and provide the computer experience which will make the use of a computer possible in later courses.

(C) Survey a variety of significant uses of computers and other automata.

Content: The lectures are composed of three parts which should not be treated as completely independent. These are:

(a) Characteristics of a procedure-oriented language. This teaches how to state a well defined problem in a form such that it can be compiled and executed. Attention should be given to motivating the features included in the language as well as the restrictions and conventions encountered.

(b) Description of a computer. Here the student should obtain an understanding of the general structure of computers, of the representation of information (both numeric and non-numeric), and of a representation of machine instructions. Only enough of an order code is taught to enable the student to write simple codes for the available machine to aid his understanding of the relationship between the computer and the algorithmic language.

(c) Introduction to algorithms. The intuitive notion of an algorithm should be introduced with special care; definiteness, generality, conclusiveness, and the limitations of their finite nature being emphasized. The representation of algorithms in narrative form (including mathematical symbolism), as flowcharts, and as computer programs should be discussed. Simple algorithms are developed for several problems such as elementary numerical calculations, sorting, simulation of a random process, and symbol manipulation. Iterative and recursive algorithms are to be explained and compared. The definitions and use of functions, subroutines, and iterative procedures are dealt with throughout.

References:

PERLIS, A. J. Programming for digital computers. *Comm. ACM* 7 (Apr. 1964), 210.

ARDEN, B. W. On introducing digital computing. *Comm. ACM* 7 (Apr. 1964), 212.

MATHEMATICAL ASSOCIATION OF AMERICA, COMMITTEE ON THE UNDERGRADUATE PROGRAM IN MATHEMATICS. Recommendations on the undergraduate mathematics program for work in computing. The Committee, P.O. 1024, Berkeley, Calif., May, 1964.

ARDEN, B. W. *An Introduction to Digital Computing.* Addison-Wesley, Reading, Mass., 1962.

2. Computer Organization and Programming (2-2-3)

This course provides the student with a thorough description of the logical organization of computers. The course content will depend to a considerable extent on the available computing equipment. Nevertheless, it is important that the course as a whole go beyond any specific computer. Hardware features which are not available should be simulated whenever possible. It is vital that this course be frequently updated to reflect changing ideas of computer organization. By the use of appropriate examples, the student is first convinced of the limitations of procedure-oriented languages and the need for a more intimate knowledge of the machine. The course treats the hardware functional units such as memory, arithmetic units, and control registers, the individual machine instructions, and use of these elements in combination to produce effective programs.

Content:

(a) Examples showing the limitations of procedure-oriented languages.

(b) Representation of the abstract notion of numbers and characters in several ways including representations used in digital machines.

(c) Functional basis of various computer structures including memory devices, word structure and addressing, arithmetic units, and input-output equipment.

(d) Control units and instruction sequencing.

(e) Machine instructions, their representation in machine language and in symbolic form, and their use together with detailed flowcharts to solve simple problems.

(f) Use of index registers, base registers, and indirect addressing.

(g) Simple input-output instructions.

(h) Open and closed subroutines and their incorporation through linkages into complete programs.

(i) Interpretive routines.

(j) Assembly programs and macros.

(k) Advanced input-output instructions and editing procedures.

References:

SHERMAN, P. M. *Programming and Coding Digital Computers.* John Wiley and Sons, New York, 1962.

WEGNER, P. *Introduction to Symbolic Programming.* Griffin, 1964.

WEGNER, P. (ED.) *Introduction to System Programming.* Academic Press, New York, 1964.

GRABBE, E. M., RAMO, S., AND WOOLDRIDGE, D. E. (EDS.) *Handbook of Automation, Computation, and Control, Volume 2: Computers and Data Processing.* Wiley, New York, 1959.

[3] See Keenan, T. A. Computers and education. *Comm. ACM 8* (Apr. 1964), 206.

3. Numerical Calculus (2-2-3)

The course is intended to provide the student with a first introduction to numerical analysis, complementing his studies of beginning analysis and continuing and deepening his understanding of the analysis of computational problems, the development of algorithms for their solutions, and the efficient utilization of the computer itself. In the Laboratory portion, the student completes a substantial number of computational projects using a suitable procedure-oriented language. Emphasis is on numerical error and the understanding of the sensitivity of problems to errors, and of pitfalls in numerical analysis.

Content:

(a) Basic Concepts of Numerical Error. Significant digit arithmetic rounding procedures. Classification of error, evaluations of expressions and functions.

(b) Interpolation and Quadrature. Polynomial interpolation elements of difference calculus, Newton and Lagrange formulas, Aitken's interpolation method, quadrature formulas, Romberg integrations, numerical differentiation and the inherent error problems.

(c) Solution of Nonlinear Equations. Bisection method, successive approximations including simple convergence proofs, linearization and Newton's method, method of false-position. Applications to polynomial equations. Generalization of iterative methods for systems of equations.

(d) Linear Systems of Equations. Solution of linear systems and determinant evaluation by elimination procedures. Roundoff errors and ill-conditioning. Iterative methods.

(e) Numerical solution of ordinary differential equations Euler's method, modified Euler's method, simplified Runge-Kutta.

References:

MACON, NATHANIEL. *Numerical Analysis*. John Wiley and Sons, New York, 1963.

NIELSEN, KAJ LEO. *Methods in Numerical Analysis*. 2nd ed., MacMillan, New York, 1964.

STIEFEL, EDWARD L. *An Introduction to Numerical Mathematics*. Academic Press, New York, 1963.

MILNE, W. E. *Numerical Calculus*. Princeton U. Press, Princeton, N.J., 1949.

STANTON, R. G. *Numerical Methods for Science and Engineering*. Prentice-Hall, Englewood Cliffs, N. J., 1961.

KUNZ, K. S. *Numerical Analysis*. McGraw-Hill, New York, 1957.

SCARBOROUGH, J. *Numerical Mathematical Analysis*. Johns Hopkins U. Press, Baltimore, 1962.

MCCRACKEN, D., AND DORN, W. S. *Numerical Methods and FORTRAN Programming*. John Wiley and Sons, New York, 1964.

WILKINSON, J. H. *Rounding Errors in Algebraic Processes*. Prentice-Hall, Englewood Cliffs, N. J., 1964.

SALVADORI, M. G., AND BARON, M. L. *Numerical Methods in Engineering*. 2nd ed., Prentice-Hall, Englewood Cliffs, N. J., 1961.

HAMMING, RICHARD W. *Numerical Methods for Scientists and Engineers*. McGraw-Hill, New York, 1962.

CONTE, S. D. *Elementary Numerical Analysis: An Algorithmic Approach*. McGraw Hill, New York, 1965.

4. Information Structures (2-2-3)

The purpose of the course is to provide an organized introduction to structures of information sets which reflect the syntactic or semantic relations in the information. The generation, development, and processing of structures such as lists and trees will be developed to illustrate interrelationships of data structures. A knowledge of machine instructions and storage hierarchies sufficient for the execution of simple algorithms is presupposed. Illustrative examples are used extensively.

Content:

(a) Description of a data base and its structure. The basic concepts of functions, arrays, records, files, trees, lists, and list structures.

(b) Fixed length, variable length, and mixed records; updating and addition to records; absolute and relative referencing of a record; linkages and transformation vectors and matrices; serial and parallel files; hierarchies of storage.

(c) Sorting, searching and retrieval from files; the role of programs in the data base, their relocation and allocation of storage.

(d) Transformations from one storage medium to a second or the same; further topics on sorting and merging of files.

(e) Referencing and processing techniques based on structure list processing, content addressing, cross referencing of files, trie memories, etc.

References:

MCCARTHY, J., ET AL. *Lisp 1.5 Programmer's Manual*. MIT Press, Cambridge, Mass., 1962.

BORKO, H. (ED.) *Computer Applications in the Behavioral Sciences*. Prentice-Hall, Englewood Cliffs, N. J., 1962.

BRUMM, G. L. *A Study of Methods for Representing Data Structure*. Mitre W5570, The Mitre Corp., Nov. 1962.

KOSAKOFF, M., AND BUSWELL, D. L. Variable information processing. Preprints 1962 ACM Nat. Conf., Sept. 1962, 112.

WANG, T. L. An information system with the ability to extract intelligence from data. *Comm. ACM 5* (Jan. 1962), 16.

GREEN, B. F., JR., ET AL. Baseball: an automatic question-answerer. Proc. Western Joint Comput. Conf., May 1961.

IVERSON, K. E. *A Programming Language*. John Wiley and Sons, New York, 1962.

NEWELL, A. (ED.). *Information Processing Language-V Manual*. 2nd ed., Prentice-Hall, Englewood Cliffs, N. J., 1964.

BROOKS, F. P., AND IVERSON, K. E. *Automatic Data Processing*. John Wiley and Sons, New York, 1963.

MARKOWITZ, H. M., HANSNER, BERNARD, AND KARR, H. W. *SIMSCRIPT, A Simulation Programming Language*. Prentice-Hall, Englewood Cliffs, N. J., 1963.

IRONS, E. T. "Structural connections" in formal languages. *Comm. ACM 7*, 2 (Feb. 1964), 67–72.

PL/I: Language Specification of IBM Operating System/360. IBM File S 360-29, C 28-6571-0.

5. Algorithmic Languages and Compilers (3-0-3)

This course is designed to acquaint the student with the formal description of procedure-oriented languages and the techniques useful in the translation of algorithms written in these languages into computer programs.

Content:

(a) Formal Languages and their abstract description, in terms of alphabet and syntax. Metalanguages such as Backus normal form.

(b) Application of formal descriptions to languages such as ALGOL and FORTRAN with a discussion of the ambiguities and semantics involved.

(c) Definition of and writing compilers for simple languages including some having a nonalgebraic base.

(d) The concept of procedures with attention to replication and recursion in their use; binding and localizing of variables including the use of block structures; dynamic allocation of storage and the conveying of data objects between sections of a process.

(e) Syntactic decomposition and the theory of compilers which are syntax directed or recursively controlled.

(f) Discussion of languages for particular application areas including other than algebraic languages.

(g) Arrangement of languages into hierarchies.

References:

GRABBE, E. M., RAMO, SIMON, AND WOOLDRIDGE, D. E. (EDS.) *Handbook of Automation, Computation, and Control*. John Wiley and Sons, New York, 1958.

BAUMANN, R., FELICIANO, M., BAUER, F. L., AND SAMELSON, K.

Introduction to ALGOL. Prentice-Hall, Englewood Cliffs, N. J., 1964.

ACM COMPILER SYMPOSIUM. *Comm. ACM 4* (Jan. 1961), 3–84.

ROSEN, SAUL. A computer-building system developed by Booker and Morris. *Comm. ACM 7* (July 1964), 403–414.

RANDALL, B., AND RUSSELL, L. J. *ALGOL 60 Implementation.* Academic Press, New York, 1964.

6. Logic Design and Switching Theory (3-0-3) or (2-2-3)

The purpose of this course is to acquaint the student with the logical design of computer elements and systems. The course can be strengthened by demonstrations or laboratories.

Content:

(a) Boolean algebra, both set theoretic and axiomatic approach, manipulative rules, propositional logic, truth tables, simplification including the use of Karnaugh maps, matrix representation and methods.

(b) Application of Boolean algebra to switching, switches and relays, tube and solid-state devices, design of combinational circuits, examples of adders and code converters.

(c) Reduction to Boolean form of algorithmic or other descriptions of processes.

(d) Basic notions of error detecting and correcting codes, majority logic, and examples of these.

(e) Logic of storage elements, basic sequential circuits, digital systems design principles.

References:

ARNOLD, B. H. *Logic and Boolean Algebra.* Prentice-Hall, Englewood Cliffs, N. J., 1962.

BARTEE, T. C. *Digital Computer Fundamentals.* McGraw-Hill, New York, 1962.

BRAUN, E. L. *Digital Computer Design.* Academic Press, New York, 1963.

CALDWELL, S. H. *Switching Circuits and Logical Design.* John Wiley and Sons, New York, 1958.

CHU, Y. *Digital Computer Design Fundamentals.* McGraw-Hill, New York, 1962.

TORNG, H. C. *Introduction to the Logical Design of Switches Systems.* Addison Wesley, Reading, Mass., 1964.

BUCHHOLZ, W. *Planning a Computer System.* McGraw-Hill, New York, 1962.

FLORES, I. *Computer Logic.* Prentice-Hall, Englewood Cliffs, N. J., 1960.

MCCLUSKEY, E. J., AND BARTEE, T. C. *A Survey of Switching Circuit Theory.* McGraw-Hill, New York, 1962.

PHISTER, MONTGOMERY, JR. *Logical Design of Digital Computers.* John Wiley and Sons, 1958.

7. Numerical Analysis I (3-0-3)

This course is intended to give a rigorous treatment of the following topics:

(a) Solution of Equations. Newton's method and other iterative methods for solving systems of equations; Aitken's Δ^2 process; Newton-Bairstow method. Muller's method, and Bernoulli's method for polynomial equations; convergence and rates of convergence are discussed for each method.

(b) Interpolation and Approximation. Polynomial interpolation; Lagrange's method with error formula; Gregory-Newton and other equal interval interpolation methods; systems of orthogonal polynomials; least squares approximation; trigonometric approximation; best approximation in the Chebyshev sense.

(c) Numerical Differentiation and Quadrature. Formulas involving equal intervals; Romberg integration, extrapolation to the limit; Gaussian quadrature.

(d) Solution of Ordinary Differential Equations. Runge-Kutta methods; multistep methods; predictor-corrector methods; stability.

8. Numerical Analysis II (3-0-3)

(a) Linear Algebra. Rigorous treatment of elimination methods and their use to solve linear systems, invert matrices, and evaluate determinants; compact schemes; methods for solving the eigenvalue-eigenvector problem including the power method, the inverse power method, Jacobi's method, Givens' method and Householder's method; roundoff analysis, conditioning.

(b) Numerical solution of boundary value problems in ordinary differential equations.

(c) Introduction to the numerical solution of partial differential equations. The emphasis is primarily in the computational aspects of the use of finite difference methods for linear equations; determination of grids; derivation of difference equations; the solution of large linear systems by iterative methods such as simultaneous displacements, successive displacements, and successive over-relaxation; norms and spectral radii of matrices and their use in the study of convergence properties of various iterative methods; applications to linear systems arising from elliptic equations.

References:

See the references listed after course 3.

9. Computer and Programming Systems (3-0-3)

Selections for a single semester course can be made from the material listed below.

Contents:

(a) Computer-Oriented Programming Systems
- (i) Design of assembly systems
- (ii) Macro-instructions and their generalizations, conditional and recursive macro-instructions
- (iii) Structure of program libraries
- (iv) Program intercommunication, linking and symbolic references between programs at load time
- (v) Input-output programming systems
- (vi) Debugging systems, source language debugging at load time
- (vii) Meta-assembly systems and other approaches to languages for writing software
- (viii) Batch processing executive systems
- (ix) Concept of a demand system

(b) Multi-programming and Multi-processor Systems
- (i) Interrupt systems
- (ii) Sequential multi-programming
- (iii) Storage protection
- (iv) Hardware aided multi-programming
- (v) Priorities and scheduling
- (vi) Dynamic allocation and reallocation of storage
- (vii) On-line console time sharing systems
  - (1) Man-machine communication
  - (2) Special language requirements
  - (3) Online debugging
- (viii) Use of Bulk Memory Devices for program storage
- (ix) Storage of programs in intermediate language or in partially translated form
- (x) Use of optical input and output devices
- (xi) Simple two-processor system in which one is a peripheral processor
- (xii) Multiple processors with shared main memory and/or shared peripheral devices
- (xiii) Multi-computer communication handling and other online real-time systems
- (xiv) Programming system and language requirements in multi-programming and multi-processor systems

References:

Systems manuals and various reports in Proceedings of Joint Computer Conferences.

MEALY, GEORGE H. Operating systems. Doc. P-2584, RAND Corp., Santa Monica, Calif.

WEGNER, P. (ED.) *Introduction to System Programming.* Academic Press, New York, 1964.

10. Combinatorics and Graph Theory (3-0-3)

The purpose of this course is to provide an introduction to set theory, graph theory, and combinatorial analysis.

Content:

(a) Set Theory
- (i) Set algebra
- (ii) Product sets
- (iii) Relations and functions
- (iv) Order relations
- (v) Cardinality
- (vi) Isomorphism

(b) Graph Theory
- (i) Basic definitions and concepts
- (ii) Undirected graphs
  - (1) Trees
  - (2) Graphs with cycles
- (iii) Directed graphs
- (iv) Operations on graphs

(c) Combinatorial Analysis
- (i) Elementary combinatorics, binomial coefficients
- (ii) The principle of inclusion and exclusion
- (iii) Recurrence relations
- (iv) Systems of distinct representatives
- (v) Matrices of zeros and ones
- (vi) Partition and compositions
- (vii) Enumeration and Polya's Theorem

References:

BERGE, CLAUDE. *Theory of Graphs and Its Applications.* John Wiley and Sons, New York, 1962.

HALL, MARSHALL. A survey of combinatorial mathematics. In I. Kaplansky, et al., *Some Aspects of Analysis and Probability.* Surveys in Appl. Math. No. 4, John Wiley and Sons, Inc., New York, 1958.

KONIG, DENES. *Theorie der endlichen und endlichen Graphen, Kombinatorische Topologie der Streckenkomplexe.* Chelsea Publishing Co., New York, 1950.

FLAMENT, CLAUDE. *Applications of Graph Theory to Group Structure.* Prentice-Hall, Englewood Cliffs, N. J., 1963.

ORE, OYSTEIN. *Graphs and Their Uses.* Random House, New York, 1963.

RYSER, HERBERT J. *Combinatorial Mathematics.* John Wiley and Sons, Inc., New York, 1963.

RIORDAN, JOHN. *An Introduction to Combinatorial Analysis.* John Wiley and Sons, Inc., New York, 1958.

11. Systems Simulations (2-2-3)

This course gives an introduction to the development of logical, numerical and statistical models of systems. Here the computer is used to carry out actual experiments rather than to compute results based on predetermined analysis.

Content:

(a) Basic elements involved in simulation such as entities, attributes, events, actions, facilities, storage, clock time, queues, routing, priority and their interrelation.

(b) A review of sampling theory and statistical inference; random number generation, and simple queuing theory.

(c) Properties of languages useful in the description of models including the introduction and use of specific languages such as DYNAMO, GPSSII and SIMSCRIPT.

(d) Design of models including selection of scope, identification of exogenous and endogenous events, generation and termination of entities, establishment of facilities and queues, synchronization, and collection of information.

(e) Development and testing of models including queuing models, feed-back systems, storage systems, priority systems and flow in complex networks. Comparison of various designs and optimization of parameters.

References:

TOCHER, K. D. *The Art of Simulation.* D. Van Nostrand and Co., New York, 1963.

MARKOWITZ, HOUSNER, AND KARL. *SIMSCRIPT: A Simulation Language.* Prentice-Hall, Englewood Cliffs, N. J., 1962.

NATIONAL BUREAU OF STANDARDS. *Monte Carlo Methods.* NBS Appl. Math. Series 12, US Government Printing Off., Washington, D. C., 1951.

Symposium on Monte Carlo Methods. (U. of Florida) John Wiley and Sons, Inc., New York, 1956.

GORDON, GEOFFREY. General Purpose System Simulator II. IBM Corp., B2-6346, 1963.

MCMILLAN, C., AND GONZOLEZ, R. F. *Systems Analysis.* Irwin, Inc., 1965.

CHORAFAS, D. H. *Systems and Simulation.* Academic Press, New York, 1965.

12. Mathematical Optimization Techniques (3-0-3)

Optimization problems based on similar types of mathematical models arise in the extremal problems of science and engineering, in the analysis of complex systems, and in the solution of single-stage and multi-stage decision problems in management science. Mathematical optimization in these various areas is closely associated with computer science because recently developed methods have, for the most part, been meaningful only in conjunction with computer systems and programs, and through computational experimentation to determine the merits of proposed techniques.

The computer science course in this area should deal primarily with algorithms (and their mathematical foundations) for optimization of determinate models. It should cover classical methods for multivariate extremal problems as well as recent developments such as linear and dynamic programming. It should examine the computability limitations of the methods which are presented as well as their capabilities. Development of flowcharts and/or computer programming by students is recommended. A substantial course in numerical analysis is a prerequisite to this course.

Contents. Constrained extrema, Lagrange multipliers, gradient methods, optimization properties of convex function, the simplex method (linear programming), a quadratic programming algorithm, dynamic programming.

References:

BALAKRISHNAN, A. V., AND NEUSTADT, L. W. (EDS.) *Computing Methods in Optimization Problems.* Academic Press, New York, 1964.

DANTZIG, G. B. *Linear Programming and Extensions.* Princeton U. Press, Princeton, N. J., 1963.

BELLMAN, R. E., AND DREYFUS, S. E. *Applied Dynamic Programming.* Princeton U. Press, Princeton, N. J., 1962.

HADLEY, G. *Linear Programming.* Addison-Wesley, Reading, Mass., 1962.

LEITMANN, G. (ED.) *Optimization Techniques with Applications to Aerospace Systems.* Academic Press, New York, 1962.

13. Constructive Logic (3-0-3)

The purpose of this course is to provide a basic knowledge of logic and Boolean algebra for the computer sciences. If course 6 is used as a prerequisite, then some of the topics below can be eliminated to avoid duplication.

Content:

(a) Introduction
- (i) Set Algebra
- (ii) Product Sets
- (iii) Relations and Functions

(b) Boolean Algebra

(c) Propositional Calculus
- (i) Basic notation
  - (1) Standard
  - (2) Polish
- (ii) Well-formed formulas
- (iii) Truth-values and truth tables
- (iv) Normal forms
- (v) Axiomatics

(d) Algorithms
(i) Intuitive discussion
(ii) Turing machines
(iii) Unsolvable problems
(e) Quantification theory
(i) Basic notation, well-formed formulas
(ii) Satisfiability and validity
(iii) Models
(iv) First Order Theories
(v) Completeness Theorems
(vi) Mata Theorems
(vii) Equality
(viii) Extension of first order theories
(ix) Normal forms
(x) Foundational considerations

References:

TRAKHTENBROT, B. A. *Algorithms and Automatic Computing Machines*. (Transl.) Heath and Company, Boston, 1963.

MENDELSON, ELLIOTT. *Introduction to Mathematical Logic*. Ch. 1 and 2, Van Nostrand, Princeton, N. J., 1964.

KORFHAGE, ROBERT R. Logic for the computer sciences. *Comm. ACM 7* (Apr. 1964), 216–218.

KLEENE, S. C. *Introduction to Metamathematics*. D. Van Nostrand, New York, 1952.

DAVIS, MARTIN. *Computability and Unsolvability*. McGraw-Hill, New York, 1958.

MARKOV, A. A. *Theory of Algorithms*. (Transl.) Academy of Sciences of the USSR, 1954. Available from Office of Technical Services, US Department of Commerce, Washington, D. C.

14. Introduction to Automata Theory (3-0-3)

This course provides an introduction to automata theory. It should be given at the senior level after the student has completed such courses as constructive logic and algebraic structures.

Content:

(a) Definition of the notion of finite automata. Methods of going from one type of description to another; state tables, synchronous sequential circuits, etc.

(b) What can a finite automata do? Kleene's theorem on the representability of events.

(c) Reduced forms for sequential machines.

(d) Algebraic description of finite automata; semi-groups, partitions of semi-groups, monomorphisms; Hartmanis' partition theory on machine structure.

(e) Introductory material on generator systems. Types 0, 1, 2, and 3 grammars; introduction to algebraic linguistics.

(f) Other classes of machines: potentially infinite machines such as Turing machines (a type zero grammar) nondeterministic machines, probabilistic machines.

(g) The theory of computability. Statement of fundamental theorems on recursive functions.

(h) The existence of self-reproducing and self-repairing machines. The theory of Von Neumann and Myhill.

(i) Programming systems as considered formally by Myhill or Sheppardson and Sturgis. This gets to the core of the program concept and introduces one to partial recursive functions with a minimum of conceptual machinery.

References:

GILL, A. *Introduction to the Theory of Finite State Machines*. McGraw-Hill, New York, 1962.

GINSBURG, S. *An Introduction to Mathematical Machine Theory*. Addison-Wesley, Cambridge, 1962.

MCNAUGHTON, R. The theory of automata. In *Advances in Computers, Vol. 2*, Academic Press, New York, 1961.

SHANNON AND MCCARTHY. *Automata Studies*. Princeton U. Press, Princeton, N. J., 1956.

MOORE, E. F. *Sequential Machines*. Addison-Wesley, Reading, Mass., 1964.

15. Formal Languages

(a) Programming languages—methods of specification, Backus normal form, syntax charts, Iverson notation.

(b) Natural languages—parsing, syntactic analysis.

(c) Formal languages—types of grammars, phrase structure, production and recognition, context-sensitive, context-free, bounded context.

(d) Mathematical properties of context-free languages, ambiguity and decidability.

(e) The relation of formal languages to finite automata programming languages and natural languages.

(f) Computer Applications, syntactic analysis of statements in programming languages and in natural languages, syntax-directed compilers.

References:

CHOMSKY, N., AND MILLER. Introduction to the formal analysis of natural languages. Ch. 11 in *Handbook of Mathematical Psychology, Vol. II* (R. D. Luce, R. R. Bush, E. Galanter, EDS.), John Wiley and Sons, New York, 1963.

——. Formal properties of grammars. Ch. 12 in *Handbook of Mathematical Psychology, Vol. II* (R. D. Luce, R. R. Bush, E. Galanter, EDS.), John Wiley and Sons, New York, 1963.

RABIN, M. O., AND SCOTT, D. Finite automata and their decision problems. *IBM J. Res. Develop. 3* (Apr. 1959), 114–124.

KUNO, S., AND OETTINGER, A. G. Syntactic structure and ambiguity of English. Proc. 1963 Fall Joint Comput. Conf., Spartan Books, Baltimore, 1963.

BAR-HILLEL, Y., PERLES, M., AND SHAMIR, E. On formal properties of simple phrase structure grammars. *Z. Phonetik Sprachwiss. Kommunik.-forsch. 14* (1961), 145–172.

GREIBACH, S. A. Formal parsing systems. *Comm. ACM 7* (Aug. 1964), 499–504.

16. Heuristic Programming (3-0-3)t

This course introduces the student to some nonarithmetical aspects of electronic computing. He has become acquainted with certain simulation techniques and with the symbol-manipulating power and general purpose nature of computers in other courses. He now learns about projects which aim at either achieving goals that are considered to require human mental capabilities (Artificial Intelligence) or modeling highly organized intellectual activities (Simulation of Cognitive Behavior). The student also hears about research areas closely related to the above two.

Content:

(a) (i) Computing machines and intelligence.
(ii) Game playing programs.
(iii) Theorem proving by computers.
(iv) Symbolic integration and differentiation on computers.
(v) Natural language processors.
(vi) Pattern recognition.
(vii) Heuristic line balancing, and other projects.

(b) (i) The General Problem Solver.
(ii) Simulation of learning and concept formation.
(iii) Simulation of decision making.
(iv) Simulation of man-machine relations, of small group behavior, etc.

(c) (i) Self-organizing systems.
(ii) Neural Cybernetics.
(iii) Automata theory.
(iv) Machine translation of languages.
(v) Information retrieval.
(vi) Automatic medical diagnosis.
(vii) Machine composition of music.
(viii) Automatic engineering design, etc.

Reference:

FEIGENBAUM, E. A., AND FELDMAN, JULIAN (EDS.) *Computer and Thought*. McGraw-Hill, New York, 1963. The whole book contains useful reference material; attention is drawn to a Selected Descriptor-Indexed Bibliography in Part 4, by Marvin Minsky.