Summary Report: MAA Program Study Group on Computing and Computational Science

Henry M. Walker, Grinnell College (Chair)
Daniel Kaplan, Macalester College
Douglas Baldwin, SUNY Geneseo

Introduction and Themes

"Computing" and "computational science" are broad terms that encompass multiple disciplines and undergraduate programs. It is useful to consider a range of computing-related disciplines; indeed, this Program Study Group uses the international term "computing" rather than the specific term "computer science" in its title to reflect this diversity.

At the undergraduate level, Computing Curricula 2005: The Overview Report [1] (citations may be found in the companion Supplemental Report), by the Association for Computing Machinery (ACM) and the Computer Society of the Institute for Electronics and Electronics Engineers (IEEE-CS) with the Association for Information Systems (AIS), provide separate curricular recommendations for each of five major programs in computing:

- *Computer engineering* focuses on hardware and the physical connectivity of equipment.

- *Computer science* "spans a wide range, from its theoretical and algorithmic foundations to cutting-edge developments" and includes software development and "effective ways to solve computing problems." [1, pp. 13-14]

- *Information systems* focuses "on integrating information technology and businesses processes to meet the information needs of businesses and other enterprises". [1, p. 14]

- *Information technology* refers "to undergraduate degree programs that prepare students to meet the computer technology needs of business, government, healthcare, schools, and other kinds of organizations", typically emphasizing technology "more than on the information it conveys." [1, p. 14]

- *Software engineering* focuses on methodologies for the development and maintenance of reliable and efficient software.

In addition to these five, we consider two additional areas of computing:

- *Computational science and big data*. The ACM/IEEE-CS Computing Science Curricula 2013, Strawman Draft (February 2012) describes computational science as "the application of computer science to solve problems across a range of disciplines" [5, p. 47]. These disciplines are predominantly those in the natural sciences, but applications are increasingly in other areas.

- *Connections among statistics, machine learning, and databases* are of considerable and growing importance. A term in current use is “big data”; it denotes an emerging, interdisciplinary field that seeks to provide insights and analysis of the huge data sets that are collected in a wide range of application areas.

Mathematics interacts with each of these disciplines through a wide range of courses, interdisciplinary programs, majors, and minors. The connections between mathematics and these allied disciplines are not uniform; they depend substantially on the specific program. The examination of these connections entails considerations along at least two complementary dimensions. First, mathematics embraces at least three general perspectives: continuous mathematics, discrete mathematics, and statistics. Second, computing and computational science involve the various disciplines already described.

To clarify relationships between mathematics programs and programs involving computing and computational science, the next three sections of this Summary Report are organized according to traditional mathematical perspectives. In contrast, the structure of the associated Supplemental Report highlights illustrative programs that combine mathematics with various components of computing and computational science.

The final section of this Summary Report presents an abbreviated table that relates mathematical topics/courses (one dimension) to the various disciplines within computing and computational science (a second dimension).

Connections emphasizing continuous mathematics

Continuous mathematics is represented in the undergraduate curriculum by courses such as calculus, linear algebra, real analysis, and differential equations. The traditional delineation of topics was not established with reference to computing needs, so care must be taken to examine the sub-topics within each course, some of which are of great relevance to computing and some of which are not particularly important.

In general, the areas of computing that connect to natural science, to modeling, to operations research, and to engineering relating to physical principles have a strong need for continuous mathematics. These include scientific computing, computational science, and big data.

Computing has eroded the sharp distinction between continuous and discrete mathematics. Continuous quantities are often represented in discrete form. Important algorithms for solving, optimization, and probability use discrete steps in continuous settings. Examples are Markov Chain Monte Carlo methods for constructing probability distributions, multivariate integration in high dimensions, as well as solvers and optimizers. The computationally savvy student needs to understand connections between the continuous and discrete.

Cognitive goals. For computational science, it is important to have a solid grasp of concepts, such as equilibrium and stability, rather than algebraic techniques, such as integrating factors. The concept of function is central, as are operations such as differentiation, integration, solving, and optimization. Students need to understand a variety of representations of functions, not just algebraic formulas.

Connections emphasizing discrete mathematics

“Discrete mathematics” is a broad label for a variety of mathematical topics that share a focus on values or structures that cannot change smoothly and continuously. Common examples include logic, combinatorics, set theory, number theory, and graph theory. Many undergraduate mathematics programs include a course, titled something like “discrete mathematics, ” that is an introductory survey of some of these topics. Such courses also often provide an introduction to proof techniques appropriate for discrete problems; induction is a notable example.

The central objects in computer science and computer engineering are inherently discrete: algorithms are sequences of discrete steps; data structures contain only whole numbers of elements; computers themselves operate on two-valued bits; etc. As noted above, even continuous phenomena are often modeled algorithmically by discrete processes. Discrete mathematics is thus the core mathematics for computer science, software engineering, and computer engineering. The ACM/IEEE-CS model curriculum for computer science [1] defines a body of knowledge in discrete mathematics for undergraduate computer science programs; many other model curricula in computing adopt this body of knowledge or a close variant of it.

Cognitive goals. For both computational science and computer science, students need to understand a wide range of topics within discrete mathematics in order to reason about computation. Particularly important elements include propositional logic; the concepts of set, function, and relation; proof by [structural] induction; methods of counting via combinations, permutations, and recurrence relations; graph and tree structures; and discrete probability.

Connections emphasizing statistics

Statistics is likely the most widely used university-level quantitative and computational topic. Correspondingly, statistics is taught in a distributed way, with different disciplines teaching their own flavor. In math departments, the introductory course often emphasizes applied probability and features simple, one-variable descriptions (e.g, mean, proportion, standard deviation) and classical tests (e.g., the t-test) using an algebra- and formula-based approach.

This organization of statistics does not serve computational students well. Computational science students, like other science students, need to be able to read the scientific literature. Nowadays, this calls for an understanding of the statistics of multiple variables. [14]

The algebra and probability derivations on which typical statistics courses are based reflect pre-computer age representations and calculations; modern computing offers a different and broader approach. Computationally-intensive simulation-based techniques (permutation tests and bootstrapping) provide a better understanding of statistical principles than do plug-in formulas, and are more easily adapted to new types of data and applications. Happily, many of the same approaches to statistics that would benefit computational scientists are those advocated by statistical educators. For instance, George Cobb argues for an essentially computational approach to the logic of statistical reasoning — the "three R's" of randomization, repetition, and rejection. [15]

Cognitive goals. For computational science, an understanding of the logic of statistical reasoning, the contrast between explained and unexplained variation, ways to quantify this through confidence intervals and hypothesis testing, the importance of covariates and ways to deal with them, and the central role of sampling variation.

Summarizing table

The following table provides a high-level overview of the mathematical needs of typical undergraduates studying various areas within computing and computational science. Students specializing in certain subdisciplines may require additional background.

We use the following abbreviations. (Sources given in the Supplemental Report.)

- *CE*: Computer engineering: ACM/IEEE-CS recommendations from 2004 [2]
- *CS*: Computer Science: ACM/IEEE-CS Strawman recommendations 2012 [5]
- *IS*: Information Systems: ACM/AIS recommendations from 2010 [7]
- *IT*: Information Technology: ACM/IEEE-CS recommendations 2008 [9]
- *SE*: Software Engineering: ACM/IEEE-CS recommendations from 2004 [10]
- *C-alS*: Computational Science
- *BD*: Big Data

Links for all ACM/IEEE-CS recommendations can be found at http://www.acm.org/education/curricula-recommendations .

In the following table, C indicates conceptual mastery required, P practical mastery, and M modest practical understanding.

	CS	**CE**	**IS**	**IT**	**SE**	**C-alS**	**BG**
Discrete: Sets, relations, functions	C, P	C, P	M	C, P	C, P		C
Basic counting	C, P	C, P	M		C, P		C
Combinations, permutations	C, P	C, P	M		C, P		
Graphs, trees	C, P	C, P		C, P	C, P	C	
Logic: Propositional, predicate calculus	C, P	C, P		C, P	C, P		
Proof structure, techniques	C, P	C, P			C, P		
Structural induction	P	P			P		
Recursive definition	C, P	C, P			C, P		
Calculus: 1-variable	P	C		M		C, P	C, P
Multivariable						C, P	C, P
ODE						C, P	
PDE						C, P	
Linear Algebra: Matrices	P	C, P				C	C
Decompositions						C, P	C
Iterative methods						C	C
Eigenvalues and eigenvectors						C, P	C
Vector space theory						C	C
Probability: Finite probability	C, P	C, P	M	P	C, P	C	C, P
Conditional prob., Bayes' Thm.	C, P	C, P	M	P	C, P	C	C, P

Expectation	C, P	C, P	M	P	C, P	C	C, P
Independence	C, P	C, P	M	P	C, P		C
Variance		C	M	P			C, P
Statistics: Distributions	P		M			C, P	C, P
Inference	P		C	P		C, P	C, P
Conditioning/Adjustment			M			C, P	C, P
Hypothesis testing		C, P	M	C, P		C, P	C, P
Sampling		C, P	M	C, P			
Mathematical modeling: Architectures						C, P	C, P
Simulations				P		C, P	C