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

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

Table of Contents

Section 1: Introduction and overview

This Supplemental Report complements and expands an associated Summary Report with a goal of exploring relationships between fields within mathematics and subdisciplines within computing and computational science. Although topics in both mathematics and computing/computational science can support each other, practitioners of these subjects typically approach their disciplines and courses differently.

- Mathematicians typically organize mathematics into at least three broad categories: continuous mathematics, discrete mathematics, and stochastic mathematics. Often, common introductory courses fit within a single category (e.g., calculus fits within continuous mathematics, logic within discrete mathematics, and probability and statistics within stochastic mathematics). At an intermediate level, some courses, such as mathematical modeling, may fall within a single category or may combine ideas in multiple categories.
- Computing faculty and computational scientists identify at least seven distinct subdisciplines (software engineering, computer science, information systems, information technology, software engineering, computational science, and big data). Perspectives, emphases, approaches, techniques and uses of mathematics vary substantially from one subdiscipline to another.

With these differences of perspective in mind, the Summary Report presents a two dimensional table with mathematical topics as one dimension (rows) and computing/computational science subdisciplines as a second dimension (columns). To relate quickly and easily with a mathematical audience (MAA members), the Summary Report follows the familiar categories of continuous, discrete, and stochastic mathematics.

This Supplemental Report follows a different structure. Section 2 provides a careful definition of each of seven distinct subdisciplines within computing and computational science. Section 2 also notes that the Association for Computing Machinery (ACM), the Computer Society of the Institute for Electrical and Electronics Engineers (IEEE-CS), and the Association for Information Systems (AIS) publish separate curricular guidelines for five of these subdisciplines, in addition to an overview document.

With these subdisciplines identified, Section 3 provides some general comments regarding relationships between education in mathematics and various subdisciplines of computing and computation.

At a more detailed level, opportunities arise for cooperation and coordination with mathematics programs and each subdiscipline within computing and computation science. Possibilities include the following:

- mathematics courses to support majors in a subdiscipline
- mathematics minor paired with a major in a subdiscipline
- mathematics major paired with a minor in a subdiscipline
- tracks within a mathematics major that emphasize topics in a subdiscipline
- double majors involving both mathematics and the subdiscipline
- interdisciplinary majors

With so many opportunities between mathematics and each subdiscipline, a full discussion of each possible combination is impractical. Instead, Section 4 outlines five programs that illustrate some common ways that mathematics may connect with a subdiscipline of computing or computational science. This Supplemental Report then examines each of these illustrative programs in some detail in separate sections (Sections 5 through 9).

This Supplemental Report concludes with acknowledgments (Section 10) and a bibliography in Section 11.

Section 2: Mid-level definitions

On many campuses, several undergraduate programs have evolved to provide background related to computing and computation. However, since these fields are diverse, programs focus on selected elements of these subjects. Over the years, several names have emerged within the computing field to describe one focus or another.

To those outside the field, "computer science" may seem a generic term that covers many areas. For example, mathematicians may consider computing as a unified subject, suggesting that mathematics preparation for computing involves a single collection of recommendations. As the rest of this report indicates, however, the

mathematics needed for some areas of computing and computation are quite different than the mathematics needed for other areas.

The following definitions and explanations are intended to clarify several commonly-used terms related to computing and computation. First, the professional societies (the Association for Computing Machinery (ACM), the Computer Society of the Institute of Electrical and Electronics Engineers (IEEE-CS), and the Association for Information Systems (AIS)) identify five different foci for undergraduate programs. The following definitions are taken from Computing Curricula 2005: The Overview Report, developed by a joint ACM/IEEE-CS task force [1].

- *Computer engineering* "is concerned with the design and construction of computers and computer-based systems. It involves the study of hardware, software, communications, and the interaction among them. Its curriculum focuses on the theories, principles, and practices of traditional electrical engineering and mathematics and applies them to the problems of designing computers and computer-based devices." [1, p. 13]

- *Computer science* "spans a wide range, from its theoretical and algorithmic foundations to cutting-edge developments in robots, computer vision, intelligent systems, bioinformatics, and other exciting areas. We can think of the work of computer scientists as falling into three categories. They design and implement software. ... They devise new ways to use computers. ... they develop effective ways to solve computing problems. ... Computer science spans the range from theory through programming." [1, pp. 13-14]

- "*Information systems* specialists focus on integrating information technology solutions and businesses processes to meet the information needs of businesses and other enterprises, enabling them to achieve their objectives in an effective, efficient way. This discipline's perspective on information technology emphasizes information, and views technology as an instrument for generating, processing, and distributing information." [1, p. 14]

- *Information technology* "is a label that has two meanings. In the broadest sense, the term information technology is often used to refer to all of computing. In academia, it refers to undergraduate degree programs that prepare students to meet the computer technology needs of business, government, healthcare, schools, and other kinds of organizations. In some nations, other names are used for such degree programs." While "Information Systems focuses on the information aspects of information technology", the "emphasis [of information technology] is on the technology itself more than on the information it conveys." [1, p. 14]

- *Software engineering* "is the discipline of developing and maintaining software systems that behave reliably and efficiently, are affordable to develop and maintain, and satisfy all of the requirements that customers have defined for them. ... it has evolved in response to factors such as the growing impact of large and expensive software systems in a wide range of

situations and the increased importance of software in safety-critical applications. Software engineering is different in character from other engineering disciplines due to both the intangible nature of software and the discontinuous nature of software. It seeks to integrate the principles of mathematics and computer science with the engineering practices developed for tangible, physical artifacts." [1, p. 15]

Similarly, the term "computation" can connect with several fields of study:

- *Computational science* "(or scientific computing) is concerned with constructing mathematical models and quantitative analysis techniques and using computers to analyze and solve scientific problems. In practical use, it is typically the application of computer simulation and other forms of computation from numerical analysis and theoretical computer science to problems in various scientific disciplines." [a particularly well-phrased definition from 13]. Historically, much research has focused upon the use of computers to provide insight into problems within scientific disciplines. Some important areas include "precision of numerical representation, error analysis, numerical techniques, parallel architectures and algorithms, modeling and simulation, and scientific visualization." [3, p. 154] Since many traditional applications of computation arose in areas of science, scholarship in this area historically was called "scientific computing". However, in recent years, researchers have observed that the same issues apply in many application areas –- not just in science. Thus, many modern references use the term "computational science" for study and scholarship in this broad area.

- *Big data*: Developments in science and instrumentation as well as the information-gathering and collating capabilities of governments and non-governmental organizations have produced a flood of data. Use of these data involves a range of skills: statistics, modeling, database design and access, machine learning, and so on. The term "big data" has come into vogue to describe the area.

Section 3: Mathematics and Computing

Mathematics and computing are entwined: a symbiotic relationship. Mathematics provides vital conceptual perspectives, problem-solving approaches, and techniques for much computational science, big data, and many (but not all) areas of computing. Similarly, computational science, big data, and other areas of computing can provide insights and support for subjects within mathematics.

There are also tensions between the respective education communities. Computing professionals and mathematicians have different perspectives on mathematics—on the relative merits of pure and applied mathematics, for instance, or on the relative importance of numeric approximations to answers versus exact symbolic results. Both the synergies and the tensions have implications for teaching

computing and mathematics, particularly in areas where the disciplines intersect; this section examines several of those implications.

a. Pure vs applied mathematics

Mathematics is, of course, useful. Mathematicians as a group are inclined to emphasize the utility of mathematics for training minds in logical reasoning and proof. Within fields of computing, however, mathematics is equally if not more important as a way to envision and work with the objects involved in computation. Some of these objects are intrinsically mathematical, such as matrices, functions, graphs, differentiation, or integration. But mathematical representations also provide a framework for understanding data structures, the algorithms that work on those structures, and the problems those algorithms solve. Development of mathematical representations of and operations on such things is sometimes called "modeling," or at least is a component of the modeling process. People in computing fields emphasize this role of mathematics, yet it is one that is not always conveyed through mathematics curricula.

b. Elegance

A primary objective common to mathematics, computing, and computational science is to identify perspectives that simplify and clarify thinking about a problem. All of these disciplines value elegance. Beyond a sense of beauty and satisfaction, simple and elegant representations and perspectives encourage effective and correct ways to approach problems. In computing, for example, complexity can be a major obstacle to correct algorithms and programs, and elegance can have a direct impact on the validation and verification of systems.

Mathematics has a long history of discovering computational elegance. For example, today we take for granted abstract operations, such as addition and multiplication, yet these operations were challenging to perform until the development of appropriate representations and the corresponding algorithms: Arabic numerals and the place system, logarithms, etc. Advances in computing build on novel and changing mathematical representations of real-world objects, such as the sounds and images that today’s students take for granted as objects of computation.

c. Notation and language

Mathematics, all fields of computing, computational science, and big data require precision in communicating ideas, approaches, and techniques. For example, the steps in a proof or an algorithm must be stated unambiguously. Historically, mathematics developed as a notational foundation for the description of both logical arguments and computational procedures. These perspectives for mathematical computation often have provided a foundation for the development of computer languages and representations. More recently, computing has provided insights into mathematical theory and notation, in such areas as the lambda calculus for mathematical function theory. With the cross fertilization of notations and

formalism that underlies both mathematics and all areas of computation, some might consider computer languages as a relatively new form of mathematical notation with algebraic notation and operations as a "natural" form of computer language. Both computing and mathematics students deserve a fuller understanding of these relationships than they typically get from current curricula.

d. Traditional and modern perspectives

Over the years, mathematical education has responded inconsistently to advances in representations, algorithms, and technology. Some technology has been embraced; for example, computer- or calculator-generated line graphs are now standard in calculus instruction. On the other hand, some technology may be downplayed; for example, some techniques well suited for digital computation, such as bisection for solving equations, have not been so widely accepted. At a deeper level, some modern representations may be largely absent from the mathematics curriculum, despite their utility and evidence [18] that they provide a clear path to learning. Particularly pressing cases are Monte Carlo algorithms and bootstrapping. At the same time, some traditional but idealistic viewpoints linger, although they have limited practical utility. Integration with partial fractions, for example, relies upon factoring a polynomial, a conceptually trivial operation but often unwieldy in practice for polynomials of degree 5 or higher. Relative to computing perspectives, there is frequently an inversion of procedures. For example, students of calculus learn to optimize by solving equations, while computationally one often solves equations by optimization. Similarly, linear algebra students find eigenvalues by extracting roots of polynomials, while in practice eigenvalues are found by repeated multiplication.

Some traditional educational approaches within mathematics may yield areas of cognitive tension or dissonance for students of computing. For example, introductory students may not be taught that both traditional and computational viewpoints can provide insights to a problem:

- A traditional, symbolic algorithm may provide a conceptual but unattainable solution, while computationally-oriented representations provide a usable approximation.

- Limits may appear to be the "real" thing, while finite-differences are not, but errors in limit computations can make results unusable.

- Formal logical consequences of an analytic model (e.g. an inflection point in a growth curve) may not be justified based on the modeling inputs. However, in practice, approximate numerical representations may provide perfectly sensible optimal answers.

- Students spend years studying algebraic operations and representations (e.g., factorization), and have no way to express ideas like "randomize" and "repeat."

e. Summary

The observations in this section illustrate that many areas of computing and computational science build strongly upon traditional mathematics. Mathematics provides a foundation that may fairly be considered essential. However, modern understandings of the fields indicate that a strictly traditional view of mathematics will likely leave students underprepared for work in computing, computational science, big data, and even significant areas of mathematics. Students need to appreciate not only traditional mathematical concepts and representations but also insights and perspectives that utilize modern computational practices. Overall, the interplay of understandings in these disciplines suggests a nuanced relationship between education in mathematics and in computation. Each has much to contribute to the other today, and each can profitably influence the evolution of the other in the future.

Section 4: Illustrative types of programs

With at least seven distinct subdisciplines within computing and computational science, and many ways to combine study of the disciplines (majors, minors, joint majors, emphases, interdisciplinary majors, etc.), there are more variations than can be productively described. The next five sections of this Supplemental Report explore five illustrative programs that combine mathematics and some subdiscipline of computing or computation.

- Mathematics minor to support a computer engineering or software engineering major
- Mathematics courses to support a computer science major
- A double major in mathematics and information systems or information technology
- Mathematics major with computational science emphasis
- Mathematics major oriented toward big data

Together, these programs illustrate a range of mathematical content, sometimes at the level of supporting courses, a minor, or a major. Further, each subdiscipline of computing and computation is represented in at least one program. Taken together, these examples suggest many ways in which mathematics courses, tracks, and programs can connect with coursework in each of the seven subdisciplines of computing and computation.

Section 5: Mathematics minor to support a computer engineering or software engineering major

The ACM and IEEE-CS overview of computing curricula [1] describes two engineering fields within computing: computer engineering and software engineering. The curriculum guidelines for both fields [2, 10] are similar in their

mathematics requirements; both demand a wider exposure to mathematics than do the computer science guidelines [3, 4, 5].

Most computing fields have strong connections to mathematics, and many students find some formal study of mathematics to be an attractive complement to the study of computing. Mathematics minors fit engineering majors in computing particularly well, because the major programs are large, precluding a complete second major, but require significant study of mathematics.

a. Cognitive goals

In the words of CE 2004,

> [M]athematics provides a language for working with ideas relevant to computer engineering, specific tools for analysis and verification, and a theoretical framework for understanding important ideas [2, p. 32].

Mathematics plays a similar role with respect to software engineering. Thus an important cognitive goal for computer or software engineering majors pursuing a mathematics minor is *analytical thinking*: students should be able to identify the core issues and structures in a problem, and the core components and structures in a solution or argument, to describe those features mathematically, and to reason about them logically.

In order that mathematics serve as an effective "theoretical framework for understanding," such students should also develop *mathematical habits of mind*: students should become comfortable thinking mathematically and using mathematics to describe real-world situations.

b. Desirable mathematical goals

Computer and software engineering majors need to be able to use mathematical methods to model and analyze computing problems and systems. The computer engineering guidelines provide learning outcomes for all topics covered in the curriculum; the outcomes for mathematical topics consistently include an ability to use mathematics in problem solving and to apply it to engineering. The software engineering guidelines specify levels of mastery rather than detailed learning outcomes; they consistently require students to be able to apply mathematics to new situations and problems. Thus a mathematical outcome consistently desirable for all computer and software engineering students is

> *mathematical problem solving*: students should be able to model a problem or computing system mathematically, and to deduce properties of the problem or system from the model.

Engineers need to be able to apply mathematics to concrete problems, and computer and software engineering are no exceptions. CE 2004 explicitly calls for computer engineering students to be exposed to applications of mathematics in computer engineering as well as to its theory [2]. Another important mathematical

outcome for computer and software engineering students studying mathematics is therefore

> *adopting appropriate theory to suit applications*: mathematics courses for computer and software engineers must develop students' ability to apply mathematics to engineering problems; while theoretical foundations cannot be neglected, applications should be emphasized.

Finally, although computing draws more heavily on discrete mathematics than on continuous, undergraduate engineering programs as a whole have traditionally included a large amount of continuous mathematics. Both computer and software engineering require study of probability and statistics as well as of deterministic mathematics. A mathematics minor that complements these engineering programs thus has an opportunity to demonstrate

> *the variety of mathematical ideas*: both deterministic and stochastic mathematics should feature prominently in students' studies, and a core of discrete mathematics should be complemented with selected topics from continuous mathematics.

c. Required background for post-baccalaureate career paths

Most computer and software engineering majors pursue careers in the computing industry, although careers in research and education attract some. Most students who enter the industry eventually pursue graduate study at the masters level; those who seek careers in research and education (at the college/university level) inevitably pursue a doctorate.

Prerequisites for graduate work

Few graduate programs in computer or software engineering require specific courses, let alone specific mathematics courses, for admission. It is more common to merely require an undergraduate degree in a computing field. Among programs that do specify a required mathematics background, discrete math, statistics, and various forms of calculus are common. However, a strong mathematics background in general is very helpful to graduate study of computer or software engineering, particularly at the doctoral level. Students considering graduate school in either field should thus be encouraged to take as much mathematics as possible, and to pursue mathematics significantly beyond the introductory level. A mathematics minor is eminently appropriate for such students.

Expectations for careers in industry

Engineering development experience, particularly in a team setting, is the crucial factor in qualifying students for many first jobs in the computing industry. Mathematical background per se has little bearing, although analytical

and critical thinking, problem solving, and similar intellectual skills developed by the study of mathematics are valuable.

Beyond the entry level, analytical thinking and problem solving are important, and specific mathematical knowledge may help employees advance. In particular, most computer systems rely on mathematically motivated algorithms or models in some way, and engineers who understand and can work with the underlying mathematics are in demand.

Prerequisites for 3-2 programs
Many students, particularly from liberal arts colleges, obtain a computer engineering degree through a "3-2" program. Such programs typically involve three (or occasionally four) years of study at a liberal arts college followed by two years of study at an engineering college. Students generally receive bachelors degrees from both schools, for example, a bachelor of science from the liberal arts college and a bachelor of science in engineering from the engineering college. While such programs exist in computer engineering, they are less common in software engineering. Students may need to satisfy certain mathematics prerequisites in order to transfer to the engineering school; these requirements are typically common to all engineering disciplines, and may include some or all of the following:

- three to four semesters of calculus;
- a course in differential equations;
- a course in elementary statistics.

Computer engineering programs may add a course in discrete mathematics to these requirements. Students preparing for a 3-2 program in computer engineering are thus well served by a mathematics program that includes these courses.

d. Program-specific courses

Both CE 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering [2] and SE 2004: Software Engineering 2004 [10] specify core mathematical knowledge in discrete mathematics and probability and statistics. Important discrete mathematics topics include the following:

- *Functions, sets, and relations:* basic terminology and definitions of each and the relationships between them; the basic operations on sets

- *Logic:* primarily propositional logic and Boolean algebra, with an introduction to predicate logic

- *Proof techniques:* standard and structural induction are the most widely used proof methods in computer science and software engineering; direct proofs, proof by contradiction, proof by counterexample, and proof by contraposition are also important

- *Counting:* combinations, permutations, recurrence relations, summations, the pigeonhole principle, the rules of sum and product

- *Graphs and trees:* basic terminology and definitions, standard algorithms such as traversals and searches.

Relevant topics in probability and statistics include these:

- *Discrete probability:* basic definitions and laws for finite probability spaces, common distributions, conditional probability and Bayes's theorem, expected value, mean and standard deviation

- *Simple hypothesis testing*: null hypotheses, t-test, chi-squared test

- *Estimation:* kinds of estimate and estimator, confidence intervals

The computer engineering, but not software engineering, guidelines also call for some amount of continuous mathematics, including calculus and continuous probability.

Most mathematics minor programs would have little difficulty accommodating these requirements. Mathematics minors typically require more than enough calculus (CE 2004 [2] merely specifies "differential and integral calculus"; it is unclear whether this is meant to include multivariable calculus, but it certainly goes no farther). Many minors have at least an option of including a probability and statistics course, as an elective if in no other way. Minors that require a discrete mathematics course are rare, but most minors would allow one as an elective. Other mathematics courses that commonly appear in minors, such as linear algebra or differential equations, are helpful to computer and software engineering students even if not seen as essential to the fields.

e. Sample programs

Two example programs illustrate the variety of contexts within which a mathematics minor can serve the needs of computer or software engineering students.

M.I.T. offers a highly unrestrictive minor, specifying only that students complete six mathematics courses past an institution-wide single- and multivariable calculus requirement. The six courses must be "of essentially different content," and four must be at an advanced level. It would be quite easy within these constraints to take the discrete mathematics and probability/statistics courses expected for computer or software engineering, and to fill the remainder of the minor with either more detailed study of mathematics for computing, or with a spectrum of continuous math courses as included in other engineering programs. [19]

The *University of Rochester* program is much more prescriptive, requiring students to take two semesters of calculus and a combined linear algebra and differential

equations course for admission to the minor, a more advanced linear algebra course, either discrete mathematics or multivariable calculus, and finally two elective mathematics courses. However, even this program readily accommodates the needs of computer and software engineering: discrete mathematics satisfies one of the requirements of the minor, and a probability or statistics course can serve as one of the electives. [20]

Section 6: Mathematics courses to support a computer science major

Almost every computer science major program requires one or more mathematics courses in addition to computer science courses. The ACM and IEEE-CS curricular guidelines for computer science spell out specific mathematical foundations for undergraduate majors in those fields [3, 4, 5]. These guidelines specify both topics that students should study and the level of mastery that they should achieve. As a general rule, mathematics requirements for computer science are less demanding than those for computer or software engineering [2, 10].

a. Cognitive goals

Broadly, computational and mathematical styles of thought overlap to a significant degree. Of course, algorithms are central to computational thinking, and weave through much mathematical thinking; both the creative insight and methodical logic that go into a mathematical proof or derivation are also present in the creation and analysis of a computer algorithm or program. Thus an important cognitive goal for computer science majors studying mathematics is

> *mathematical habits of mind*: students should become comfortable thinking mathematically and appreciate such a mode of thought as one with strong parallels to algorithmic thinking (and vice versa).

Most of the specific mathematical topics computer science majors study have applications in modeling and analyzing problems to be solved computationally, or in modeling and analyzing algorithms that solve those problems. Thus another important cognitive goal for mathematics courses intended to support computer science majors is

> *analytical thinking:* students should be able to identify the core issues and structures in a problem, and the core components and structures in a solution or argument, to describe those features mathematically, and to reason about them logically.

b. Desirable mathematical goals

Computer science majors need to be able to use mathematical methods to model and analyze computing problems. For example, CC 2001: Computer Curriculum 2001 [3] includes such learning outcomes as "model problems in computer science

using graphs and trees," "analyze a problem to create relevant recurrence equations or to identify important counting questions," and "Use formal logic proofs and logical reasoning to solve problems such as puzzles." In other areas the computer science learning outcomes call only for students to be able to describe or identify mathematical ideas (e.g., "describe how formal tools of symbolic logic are used to model algorithms and real life situations"). In all cases, however, the underlying mathematical outcome is

> *mathematical problem solving*: students should be able to model a problem or algorithm mathematically, and to deduce properties of the problem or algorithm from the model.

Computer science draws on both applied and theoretical mathematics. Applied mathematics is a day-to-day tool in analyzing algorithms, while many advances in algorithm design are deeply rooted in theoretical mathematics — modern cryptography with its roots in number theory is an example. Thus another mathematical outcome for computer science and software engineering students studying mathematics is

> *adopting appropriate theory to suit applications*: mathematics courses that support computer science programs have ample opportunities to both demonstrate applications of mathematics and to explore the theoretical roots from which those applications spring.

c. Required background for post-baccalaureate career paths

Most computer science majors see themselves destined either for careers either in the computing industry or in computing research and education. Many students who enter the industry eventually pursue graduate study at the masters level; those who seek careers in research and college-level education inevitably pursue a doctorate.

Prerequisites for graduate work

Particularly at the doctoral level, a strong mathematics background is very helpful to graduate study of computer science. Some programs even find a strong mathematics background is better preparation than a strong computing background with little mathematics. Students considering graduate school in any computing field should thus be encouraged to take as much mathematics as possible, and to pursue mathematics significantly beyond the introductory level.

Expectations for careers in industry

Software development experience, particularly in a team setting, is the crucial factor in qualifying students for many first jobs in the computing industry. Mathematical background *per se* has little bearing, although analytical and critical

thinking, problem solving, and similar intellectual skills developed by the study of mathematics are valuable.

Beyond the entry level, analytical thinking and problem solving are important, and specific mathematical knowledge may help employees advance. In particular, most computer systems rely on mathematically motivated algorithms or models in some way, and senior developers or technical managers who understand and can work with the underlying mathematics are in demand.

d. Program-specific courses

CC 2001: Computer Curricula 2001 [3] and CS 2008: Computer Science Curriculum Update 2008 [4] specify core mathematical knowledge that amounts to a survey of discrete mathematics, including these ingredients:

- *Functions, sets, and relations:* basic terminology and definitions of each and the relationships between them; the basic operations on sets
- *Logic*: primarily propositional logic and Boolean algebra, with an introduction to predicate logic
- *Proof techniques*: standard and structural induction are the most widely used proof methods in computer science and software engineering; direct proofs, proof by contradiction, proof by counterexample, and proof by contraposition are also important
- *Counting*: combinations, permutations, recurrence relations, summations, the pigeonhole principle, the rules of sum and product
- *Graphs and trees*: basic terminology and definitions, standard algorithms such as traversals and searches
- *Discrete probability*: basic definitions and laws for finite probability spaces, conditional probability and Bayes's theorem, expected value.

This mathematical core is adopted verbatim or nearly verbatim by many other ACM and IEEE-CS computing curriculum volumes.

As already mentioned, the computer science curriculum guidelines ask that students be able to solve problems with selected key topics in this list, while merely being familiar with the rest. These requirements can be met with either a single course or a two-course sequence.

Beyond the core mathematics outlined above, almost any further study of mathematics is beneficial to computer science and software engineering students. The following courses are likely to be particularly pertinent:

1. *Calculus*: some calculus is often a prerequisite to further study of mathematics; differential, integral, and multivariable calculus are all relevant to some areas in computer science
2. *Linear algebra*: vectors and matrices are models for many phenomena in computer science, and are heavily used in computer graphics, information retrieval, etc.
3. *Probability, introductory statistics*: much of what happens in actual computer systems is stochastic in nature, and many algorithms and data structures exploit "randomness" in some form; statistical analysis of empirical data is valuable in all computing disciplines.

e. Sample programs

The ACM Special Interest Group on Computer Science Education has published a collection of models for covering the ACM/IEEE-CS computer science discrete mathematics requirements in a one-semester course. These models can be found in [12].

From a more focused perspective, the Liberal Arts Computer Science Consortium has described a complementary model curriculum in computer science for liberal arts programs. This liberal arts model provides details of two alternatives for a two-semester discrete mathematics sequence [17].

Section 7: A double major in mathematics and information systems or information technology

Information systems and information technology both involve the collection, storage, and processing of information for businesses, commercial ventures, governmental agencies, and other organizations. The disciplines differ, however, in their focus. As the 2005 ACM/IEEE-CS Overview Report observes, "Information Systems focuses on the information aspects of information technology. Information Technology is the complement of that perspective; its emphasis is on the technology itself more than on the information it conveys" [1, p. 14]. Organizational aspects of the enterprise also are important in these disciplines –- particularly in information systems.

Historically, curricula for information science have included relatively little mathematics. However, the most recent recommendations from ACM/AIS in 2010 [7] may suggest opportunities for expanded cooperation and integration of topics. In particular, "Mathematical Foundations" are identified as an important supporting area. Details are thin, but the report discusses mathematical needs within some areas of information systems:

> *Mathematical foundations.* Even though IS professionals do not need the same level of mathematical depth as many other computing professionals,

> there are, however, some core elements that are very important for IS professionals (of course, these needs will vary depending on an individual's specialty). To support in-depth analysis of data, IS professionals should have a strong background in statistics and probability. For those who are interested in building a strong skill set in algorithmic thinking, discrete mathematics is important. [7, p. 21.]

Within information technology, the 2008 guidelines provide reasonable detail regarding the mathematical and statistical background needed by all graduates intending to work in the information technology field. The high-level outline begins as follows:

> Mathematics and Statistics for IT (38 core hours)
> MS. Basic Logic [minimum 10 hours]
> MS. Discrete Probability [minimum 6 hours]
> MS. Functions, Relations, and Sets [minimum 6 hours]
> MS. Hypothesis Testing [minimum 5 hours]
> MS. Sampling and Descriptive Statistics [minimum 5 hours]
> MS. Graphs and Trees [minimum 4 hours]
> MS. Application of Mathematics and Statistics to IT [minimum 2 hours] [9, p. 94 ff]

The 2008 report observes that various areas within information technology may require additional mathematical and statistical background:

It is useful to point out that this knowledge unit only specifies core learning outcomes, i.e., learning outcomes that any graduate from an IT program, independent of specialization, must have acquired. Depending on their specialization, graduates may need to obtain additional knowledge of mathematics or statistics. For example, students specializing in the areas of networking or platform technologies are likely to need to become comfortable with a range of concepts from calculus, whereas students specializing in the area of information management and/or knowledge management are likely to require a deeper knowledge of statistics than this unit specifies. However, since the model curriculum is agnostic about the type of specializations that institutions create, it seems inappropriate to specify more advanced learning outcomes in detail. [9, p. 95]

Taken together, the most recent curricular guidelines for information systems and information science highlight the need for some mathematical background, particularly in the areas of discrete mathematics, probability, and statistics. In addition, the national recommendations for these disciplines provide opportunities for conversations with mathematics programs to coordinate work in double majors to strengthen students' background in significant and relevant ways.

a. Cognitive goals

Both information systems and information technology generally have a practical orientation. Typical questions might include the following:

- How can information be gathered, processed, and reported to support the functions of the business or organization?
- How can information be used to assist in decision making?
- How can cost-effective technological systems be designed, created, and maintained to support organizational needs for information?
- When systems malfunction, how can difficulties be identified and resolved effectively?

Answers to such questions require practical skills involving analytical thinking, problem solving, evidence-driven decision making, and effective communication. That is, traditional high-level perspectives, insights, and skills from the mathematical sciences can provide substantial insights and capabilities within both information systems and information technology.

> *Mathematical thinking skills:* Double majors in mathematics and either information systems or information technology should practice the mathematical skills of analytical reasoning, critical thinking, problem solving, creativity, curiosity, and communication within the context of information.

Information systems and information technology also provide a natural context to apply mathematical theory, particularly in the areas of probability and statistics and perhaps discrete mathematics.

> *Discrete and stochastic perspectives*: Since information systems and information technology are inherently practice-oriented, double majors with mathematics should be able to connect the theory that is often developed within mathematics to the practice and application in these information-oriented disciplines. Particular areas for study should involve discrete mathematics and stochastic mathematics.

b. Desirable mathematical goals

The ACM/IEEE-CS recommendations for information technology state explicit "Core learning outcomes" for "Mathematics and Statistics for IT (38 hours)" in the areas of "Basic Logic", "Discrete Probability", "Functions, Relations, and Sets", "Hypothesis Testing", "Sampling and Descriptive Statistics", "Graphs and Trees", and "Applications of Mathematics and Statistics for IT" [9, pp. 95-97]. For each topic, the stated outcomes are quite practical. For example, the following description is given for applications:

> MS. Application of Mathematics and Statistics to IT
> Minimum core coverage time: 2 hours

Topics: Math, statistics and probability in IT

Core learning outcomes:

1. Explain, with examples, the importance of a range of mathematical concepts, including sets, relations, functions, basic logic, and graphs and trees for IT.
2. Explain, with examples, the role of probability and statistics in IT.
3. Perform a statistical analysis of a system's performance.
4. Analyze a statistical analysis of a system's performance and recommend ways to improve performance. [9, p. 97]

Other stated outcomes for mathematics to support information technology are equally practical.

As already noted, the total time specified within the ACM/IEEE-CS curricular recommendations for topics in mathematics and statistics is "38 core hours" [9, p. 95] — about one typical semester-long course. With the range of topics prescribed, coverage can be at only a moderate level of depth for a standard major in information technology. Presumably, a double major would have the opportunity to explore many of these topics in considerably more depth.

Curricular recommendations regarding mathematics within information systems are not clearly delineated, but comments suggest the usefulness of discrete mathematics, probability, and statistics. Again, a double major with mathematics could greatly strengthen a student's background and add several dimensions for future professional work.

c. Required background for post-baccalaureate career paths

Undergraduate programs in both information systems and information technology generally emphasize training for careers in business, industry, and other organizations. The programs are designed to provide students with needed background and knowledge to obtain jobs and to build successful careers. From this perspective, the bachelor's degree largely is viewed as a professional credential for employment, rather than as a step toward an advanced degree.

Prerequisites for graduate work

The national guidelines for a graduate program in information systems includes this statement:

> STUDENT BACKGROUNDS
> For the foreseeable future, it is anticipated that MS programs will continue to attract students with a wide range of backgrounds. In traditional graduate programs, it is assumed that entering students have a common background obtained through an undergraduate degree in that field. For students entering the MSIS program, this is often not the case. Although students

> entering directly from undergraduate programs may have a BS degree with a major in IS, often their degree is in computer science, business, or some other field. The MSIS program may also attract experienced individuals including IS professionals and people seeking career changes. Often this experienced group will be part-time evening students or will access the courses through a remote learning environment. The architecture of the MSIS program accommodates this wide diversity of backgrounds and learning environments. [8, pp. 133-134]

To date, no national guidelines have appeared for graduate programs in information technology, but this field also seems to draw from a wide range of undergraduate backgrounds.

With the need for analytical skills, knowledge of probability and statistics, and logical reasoning, a double major that combines mathematics with information systems or information technology seems to provide a strong foundation for advanced work. However, students with bachelor degrees in many disciplines would seem to be potential candidates for graduate work in either of these disciplines.

Expectations for careers in industry

Many graduates of programs in information systems or information technology pursue careers in industry, since these programs emphasize skills that have direct relevance to companies. For example, the following statement comes from the 2010 national curricular recommendations for information systems:

> *IS Majors*: An IS major consists of the entire model curriculum targeted for a particular career track. Students proficient at this level are prepared to enter a career in the IS field. They have competencies in basic technical areas and apply these to business processes and project management. Graduates of IS programs can work for different industries such as manufacturing, financial services, health care, and others including information technology providers of hardware, software, and services. [7, p. 15]

Similarly, the 2008 national curricular recommendations for information technology provides this context:

> The fact that *Information Technology* programs emerged to meet demand from employers has had a significant effect on the evolution of the discipline. Entry-level knowledge and skill requirements gathered from potential employers of graduates naturally translate into learning or program outcomes for graduates from Information Technology programs. [9, p. 17]

Overall, undergraduate programs in both information systems and information technology seek to prepare students to enter specific types of careers within industry. These careers include a range of specialities, and this provides some diversity within programs and graduates. However, even with variations among

specialities, a career focus is a centerpiece of these types of undergraduate programs.

d. Program-specific courses: discrete mathematics, probability, and statistics

Since the national curricular guidelines for undergraduate programs in information technology specify 38 hours of coursework in areas related to discrete mathematics, probability, and statistics, a single course would likely provide a minimal covering of topics for an undergraduate degree in information technology.

A double major with mathematics, however, opens numerous possibilities to extend the level of this background. In particular, information technology makes extensive use of such areas a probability, hypothesis testing, sampling, and descriptive statistics. In mathematics programs, such topics might be explored in reasonable depth through several statistics-based courses.

Additional background from discrete mathematics also seems appropriate; and a typical, semester-long course seems consistent with background statements made in the national recommendations for both information systems and information technology. Extensive materials for this course are found in the Report of the SIGCSE Committee on the Implementation of a One-Semester Discrete Mathematics Course [12].

e. Sample programs

At a high level, national curricular recommendations provide model curricula for programs in both information systems and information technology.

The 2008 Information Technology guidelines identify two approaches to implement curricular recommendations:

An "*Integration-first approach*" identifies nine basic courses:

- IT Fundamentals
- Programming Fundamentals
- Computing Platforms
- IT Systems
- Web Systems
- Networking
- Databases
- Human-Computer Interaction
- Information Assurance and Security [9, pp. 29-30]

A "*Pillars-first approach*" organizes topics into nine different, basic courses:

- IT Fundamentals

- Programming Fundamentals
- Fundamentals of Networking
- Fundamentals of Web Systems
- Fundamentals of Information Management
- Fundamentals of Human-Computer Interaction
- System Administration and Maintenance
- Integrative Programming
- Information Assurance and Security [9, p. 30]

The 2008 report describes each of these courses in some detail. Beyond this abstract model curriculum, several recent implementations at specific schools may be obtained by performing a Web search on "sigite curriculum implementation" (where SIGITE is the ACM Special Interest Group on Information Technology Education).

Similarly, the 2010 Information Systems guidelines [7] identify seven "Core Courses" and several "Sample Elective Courses":

Core Courses

IS 2010.1 Foundations of Information Systems
IS 2010.2 Data and Information Management
IS 2010.3 Enterprise Architecture
IS 2010.4 IS Project Management
IS 2010.5 IT Infrastructure
IS 2010.6 Systems Analysis and Design
IS 2010.7 IS Strategy, Management and Acquisition

Sample Elective Courses

Application Development
Business Process Management
Enterprise Systems
Introduction to Human-Computer Interaction
IT Audit and Controls
IS Innovation and New Technologies
IT Security and Risk Management

Within the 2010 recommendations, course details include a high-level description, specific learning objectives, a detailed listing of topics, and some discussion.

Section 8: Mathematics major with computational science emphasis

Institutional forces may unduly press toward a conception of computational science that highlights its differences from mathematics, but this section advocates the

view that computational science is an emerging intellectual branch of mathematics that is part of the broad development and evolution of mathematics and its curricula over the centuries.

a. Cognitive goals

A mathematics major with computational science emphasis can highlight how mathematics and computational science can and should influence each other, rather than focus on what mathematical background is needed for computational science.

- An appreciation for the meaning of notation and the allowable forms of operation. It is not clear whether mathematics training helps students to understand the idea of notation better, but clearly even if it does not create notationally skilled workers, mathematics helps to identify and select them.

- Knowledge of the common forms of mathematical representation of real-world settings, such as matrices, differential equations, fitted curves, splines, etc.

- Understanding of the different forms in which information is conveyed and the ways that information from different sources can be combined and transformed.

- Familiarity with basic phenomena, both field-specific as in physics, chemistry, biology, and in general, such as equilibria, stability.

b. Desirable mathematical goals

Computational scientists have to be familiar with the mathematical representation of the objects of scientific interest. They also need to be familiar with mathematical modeling: the process and concepts of constructing a representation of real-world objects.

In the continuous domain, the ideas of functions and calculus in multiple variables are essential. The implementation of function- and calculus-oriented operations on the computer requires considerable knowledge of linear algebra. Indeed, matrices are important for representing many objects of scientific interest.

Symbolic techniques, although potentially a way to develop familiarity with mathematical concepts, are not central beyond a few, very basic procedures. Since many physical processes are defined in terms of differential equations, a solid grounding in modeling with ordinary and partial differential equations is essential; symbolic solution techniques, beyond the most basic, are not.

The computational scientist's mathematical background should include fitting and approximation with linear combinations of basis functions and commonly used

transforms (e.g., Fourier transforms). It is helpful if mathematical concepts are taught at least partially in the context of objects often used in computational science (e.g., "smoothness" and "continuity" in the context of splines, convergence in the context of solution via iteration). Ideas of probability and sampling are also important: important basic distributions (e.g., Gaussian, binomial, Poisson, exponential), moments and expectation values, variance, basic statistical inference.

Although many of the natural science disciplines that the computational scientist will relate to are based in continuous mathematics, the discrete domain is also important. This is so both because scientific computing typically interrelates the continuous and discrete, and because discrete representations and information are important in many areas of science, such as Markov models, trees and graphs.

Insofar as science is engaging "big data," there will be overlap between the needs of computational science and those of big data.

c. Required background for post-baccalaureate career paths

Existing graduate programs in computational science have posted very general descriptions of the backgrounds they look for in applicants. These often include a major in some "application domain," not only in mathematics but also in such fields as biology, chemistry, engineering, finance, physics, and geology. Mathematics requirements, when stated, are those typically associated with an undergraduate engineering program: three semesters of calculus, differential equations, sometimes linear algebra.

Informal conversations with employers outside of academia tend to focus on lacunae rather than requirements. Employers are looking for workers with strong modeling and communication skills. "Problem solving skills" are also strongly desired, but it is important to note that the phrase refers not to solving the sorts of problems found in calculus texts, but to the modeling phase of problems: constructing a representation of a real-world situation in mathematical terms. Ability to work in teams is also strongly emphasized.

d. Program-specific courses

As previously mentioned, courses from the first two years of a mathematics curriculum are those typically mentioned as desirable by graduate programs in computational science. This may be surprising to many faculty and may indicate a relative lack of alignment between the conventions of mathematics programs and the needs of computational science. For example, facility with probability-related concepts and computations would seem to be strongly advantageous for computational science, yet this Program Study Group identified no graduate program that requires or even recommends a probability course.

A mathematics program that wants to engage computational science should consider the following suggested ingredients:

- A *calculus* curriculum that uses mainstream computation intensively. The emphasis should certainly not be on paper-and-pencil algorithms for performing integration but on the computation and application of calculus operations. Focus symbolic computation on polynomial and simple trigonometric, log, and exponential functions and then use fitting and approximation by these functions to carry out other forms of integration.

- A *linear algebra* course oriented around "large-scale" numerical problems, not 2x2 matrices. This should include matrix decompositions and eigenvalues computed iteratively.

- A *differential equations course* connected to modeling, the study of dynamical phenomena, and mainly numerical solution of equations.

- A *statistics* course connected to modeling.

- A *simulation-based probability* course. Such a course should emphasize probability for modeling, Bayes' Rule, and computational techniques such as MCMC, rather than the application of symbolic techniques learned in traditional calculus course.

- A *modeling* course that develops and exercises skills in the construction of mathematical representations of real-world situations.

These need not be isolated, stand-alone courses. For instance, a year-long modeling sequence built on differential equations and statistics might be preferable to separate modeling, DE, and statistics courses.

Section 9: Mathematics and big data

The study of statistics should be a central component in the mathematical training of students preparing to work with big data. In mathematics departments, statistics is sometimes seen as the junior partner in "probability and statistics." From this perspective, statistics is seen and often taught as applied probability.

This is unfortunate, especially when it comes to big data. Probability is, of course, an important mathematical area in its own right, and an important tool in statistics. But it is hardly the only tool. Quite advanced statistical work can be done with basic and intuitive notions about probability distributions.

The disadvantage for statistics of a traditional focus on probability is that it leads to an emphasis on classical statistical tests, such as the t-test or one-way ANOVA, that are of limited applicability in analyzing complex systems. The distinction between the t- and normal distributions, although a considerable intellectual

achievement, is relatively specialized knowledge when it comes to big data. The topics covered in the standard introductory statistics course — means, t-tests, chi-squared tests — were developed in the days of small data generated by lab or field experiments, not big data from instrumentation and distributed data collection.

Big data is not just about large file sizes, but about large numbers of variables. To handle big data students need to be proficient at modeling with covariates, statistical adjustment/control, issues relating to multiple comparisons, and various architectures for modeling such as generalized linear models, including especially logistic regression.

Machine learning techniques — e.g., clustering and dimension reduction — are important in big data. Students need a background in statistics that equips them to deal with the corresponding statistical issues. Important techniques include resampling, bootstrapping, randomization, and cross-validation. It is feasible and sensible to base even elementary statistics instruction on these techniques.

The object of the analysis of big data is to draw conclusions and make decisions on the basis of information contained in data. Statistics is about drawing valid conclusions from data. But decision making is different; students need a background in decision theory. They also need to be able to communicate results to decision makers. This often requires sophisticated graphical presentation.

The conclusions to be drawn from big data are often intended to guide action; thus, the ability to provide a clear evaluation of causation is important. Although the most compelling support for causal claims comes from experiment, data resulting from experiments are often not available. The caveat "association is not causation," although true, does not provide sufficient support for informed action.

The last couple of decades has seen important progress toward a causal calculus. Among the important figures is 2011 Turing Award winner Judea Pearl, who observes: "The advent of causal calculi now provides a fairly transparent understanding of the assumptions that must be undertaken to answer such questions, and this enables statisticians to address directly the problems that their customers/users have kept dormant for decades." [16]

a. Cognitive goals

Interest in big data stems from the potential to discover and use patterns that reveal the operation and structure of complicated systems. It is therefore essential to have an understanding of multivariable relationships, ways of describing such relationships from data, and ways of interpreting the mathematical relations that have been discovered. Skills relating to decision making and the description and evaluation of trade-offs are important. A nuanced understanding of causation is needed to help draw useful conclusions from observational data.

Understanding of statistical logic is essential to effective work with big data. This does not necessarily mean a deep theoretical understanding of the t-test, but a range of approaches including bootstrapping and Bayesian methods.

b. Desirable mathematical goals

Students need proficiency with mathematics relating to model fitting, both linear and nonlinear and the broad set of techniques associated with machine learning. An important component of this is numerical linear algebra, including problem conditioning, matrix decomposition (such as the singular value decomposition), and techniques relevant to large-scale problems.

The mathematics relating to decision making includes constrained optimization, conditional probability, classification, and Bayesian inversion of probabilities. Topics not traditionally emphasized in mathematical approaches to statistics are important, including causal inference, structural equation modeling, matching and propensity scores, instrumental variables.

Computational science students need to be adept at the construction and statistical interpretation of simulations. Indeed, students in other computing areas benefit from being able to interpret the results of simulation and understanding the nature of variation, e.g., variation in network demand.

Careful thought should be given to preparing students to think about the world multi-dimensionally and empirically. It is inefficient, and perhaps even counter-productive, to start with a year-long introductory calculus sequence that leaves students with the idea that functions have one input and that integration and differentiation are operations applied to formulas. Students need to be able to perform operations on a range of types of objects, including not just formulas but also samples.

c. Required background for post-baccalaureate career paths

At present, facility with databases and statistical programming appear to be the rate-limiting factors in access to post-baccalaureate career paths. Communications and modeling skills are very important.

d. Program-specific courses

There is considerable overlap with the courses that support computational science, with a stronger emphasis on statistics and probability and less need for differential equations and dynamics. Linear algebra, especially as applied to large scale problems, remains central.

Section 10: Acknowledgments

The Program Study Group on Computing and Computational Science is grateful for the insights and feedback it has received from many sources. The authors greatly appreciate the interest and efforts of all people involved in the development and evolution of this work.

Presentations outlining drafts of both the Summary Report and the Supplement Report for this Program Study Group were made during sessions of the Iowa Section of the Mathematical Association of America and the Iowa Undergraduate Computer Science Consortium. Many thanks to all who attended these sessions and offered feedback.

Drafts of these reports also were shared with various groups and individuals, including Department Liaisons in the Iowa MAA and members of the Statistics Liberal Arts Workshop (SLAW), the Liberal Arts Computer Science Consortium (LACS), the Isolated Statisticians group (ISOSTAT) within the American Statistical Association (ASA), and the Iowa Undergraduate Computer Science Consortium. The report authors want to thank all who showed their interest. Particular thanks for feedback go to Thomas Moore (Grinnell College), Shonda Kuiper (Grinnell College), Tim Hesterberg (Google), Bob Hayden (statland.org), Joy Hansen (Lawrence University), Scot Drysdale (Dartmouth College), Steve Cunningham (California State University at Stanislaus), and Patrick Bailey (Calvin College).

Section 11: Bibliography

This bibliography is organized into three main subsections:

1. Materials from professional organizations
2. Books and articles about content and teaching in the area
3. Program examples

Materials from professional organizations

The Association for Computing Machinery (ACM), together with the Computer Society of the Institute of Electrical and Electronics Engineers (IEEE-CS) or the Association for Information Systems (AIS), collaborate periodically to establish curricular guidelines. This work is structured into a general, overview volume and five detailed and discipline-specific volumes. Links to the current versions of all of these reports are available at http://www.acm.org/education/curricula-recommendations/ .

1. *CC 2005: Computing Curricula 2005: The Overview Report.* A general description of curricular recommendations for five computing disciplines

2. *CE 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering*. ACM/IEEE-CS recommendations for undergraduate programs in computer engineering

3. *CC 2001: Computer Curricula 2001*. Full ACM/IEEE-CS recommendations for undergraduate programs in computer science from 2001

4. *CS 2008: Computer Science Curriculum Update 2008*. An interim update for computer science of CC 2001

5. *CS 2013: Computer Science 2013: Strawman Report (February 2012)* A full, draft revision of CC 2001 and CS 2008; a finalized revision for undergraduate programs is expected in 2013.

6. *IS 2002: Information Systems 2002.* Full ACM/IEEE-CS recommendations for undergraduate programs in information systems.

7. *IS 2010: Information Systems 2010.* An interim update for information systems of IS 2002

8. *MSIS 2006: Information Systems 2010.* Guidelines for graduate programs in information systems

9. *IT 2008: Information Technology 2008* Full ACM/IEEE-CS recommendations for undergraduate programs in information technology

10. *SE 2004: Software Engineering 2004* Full ACM/IEEE-CS recommendations for undergraduate programs in software engineering

11. *GS2E 2009: Software Engineering 2009* Guidelines for graduate programs in software engineering

As a separate project, a Committee under the ACM Special Interest Group on Computer Science Education (SIGCSE) published an extensive report on the implementation of a discrete mathematics course:

12. *Report of the SIGCSE Committee on the Implementation of a One-Semester Discrete Mathematics Course*, http://www.sigcse.org/resources/reports/discrete, April 2007.

Books and articles

13. Wikipedia, "Computational Science", http://en.wikipedia.org/wiki/Computational_science, retrieved November 14, 2012

14. Horton, N and Switzer, S, "Statistical Methods in the Journal," The New England Journal of Medicine, v.353:1977-1979, 2005

15. Cobb, G, "The Introductory Statistics Course: A Ptolemaic Curriculum" Technology Innovations in Statistics Education, 1(1), 2007, Retrieved from: http://escholarship.org/uc/item/6hb3k0nz

16. Pearl, J. "Turing Award Winner, Longtime ASA Member Talks Causal Inference" AMSTAT News No. 425, 2012

17. Liberal Arts Computer Science Consortium, "A 2007 model curriculum for a liberal arts degree in computer science", Journal of Educational Resources in Computing, 2007, Vol. 7, No. 2, Article 2.

18. Pfannkuch, M. et al. "Bootstrapping students' understanding of statistical inference, Teaching and Learning Research Initiative report, 2013 http://www.tlri.org.nz/sites/default/files/projects/9295_summary%20report_0.pdf

Program examples

18. Massachusetts Institute of Technology, "Minor in Mathematics" in "Department of Mathematics," http://web.mit.edu/catalog/degre.scien.mathe.html, accessed November 11, 2012

19. University of Rochester, "The Minor in Mathematics," http://www.math.rochester.edu/undergraduate/degrees/minor.html accesse d November 11, 2012.