ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 10, October 2016

The Importance of Mathematics in Sciences (Case Study: Computing/Computer Science)

James Vivian Chinenye, Ebiesuwa Seun, Adegbenjo. A.A, Adeyeye. J.A, Kehinde. D.O, Grace Mensah-Agyei

Department of Computer Science, Babcock University, Ilisan-Remo, Ogun State, Nigeria

Department of Computer Science, Babcock University, Ilisan-Remo, Ogun State, Nigeria

Department of Computer Science, Babcock University, Ilisan-Remo, Ogun State, Nigeria

Department of Nutrition and Dietetics, Babcock University, Ilisan-Remo, Ogun State, Nigeria

Department of Basic Sciences, Babcock University, Ilisan-Remo, Ogun State, Nigeria

Department of Microbiology, Babcock University, Ilisan-Remo, Ogun State, Nigeria

ABSTRACT: Mathematics is the abstract science of number, quantity, and space, either as abstract concepts (pure mathematics), or as applied to other disciplines such as physics and engineering (applied mathematics). Computing, finance, business, etc., should be included in this definition, in fact I think sciences and commercial should be the key definition. So in my definition I would say mathematics as an easy theoretical or conceptual science of digits, quantity, and space, either as abstract (pure mathematics), or as applied (applied mathematics) to other disciplines such as physics, engineering, computing, geography, etc., (sciences) and business administration, auditing, finance, (commercial). Mathematics may be defined as "the study of relationships among quantities, magnitudes and properties, and also of the logical operations by which unknown quantities, magnitudes, and properties may be deduced" (Microsoft Encarta Encyclopedia), Historically, it was regarded as the science of quantity, whether of magnitudes (as in geometry) or of numbers (as in arithmetic) or of the generalization of these two fields (as in algebra). Some have seen it in terms as simple as a search for patterns (algorithm). We would consider mathematics its significance, benefits and deficiencies in computing sciences and engineering in Nigeria.

KEYWORDS: Applied Mathematics; Computing; magnitude; properties; algorithm

I. INTRODUCTION

Arguably according to google computer science is the study of the principles and the use of computer, Wikipedia on the other hand defined computer science as the scientific and practical approach to computation and its applications, the systematic study of the feasibility, structure, expression, and mechanization of the
methodical procedures (or algorithms) that underlie the acquisition, representation, processing, storage, communication of, and access to information and an alternate, more succinct definition of computer science is the study of automating algorithmic processes that scale.

Presently computer science as a course name is overrated, the course is now called computing science. The questions asked will be, is Computer/Computing Science related to mathematics? Does all computing science require mathematics? Looking at these frequently asked questions from youths aspiring to go into college, we would take it one at a time.

To make it a bit comprehensible every specialization or aspect of computing has its mathematical aspect and modelling and I mean every specialization, ranging from the external hardware to the design to the function to the algorithm to the codes to artificial intelligence, cryptography, databases, to networking and as many as you can lay your hand on. When we say all aspect, we basically mean its study for knowledge and exploration, we know anybody can learn web designing, and web designs on the other hand requires some knowledge of mathematics. Sadly, but truthfully mathematics is seen as a problem especially amongst the young one, the ones who are supposed to be our future leaders, don't get me wrong, I'm not saying one can't still be a leader with little or no knowledge of mathematics, but in the context of computing, mathematics should not be overlooked. This is why we have the Indians, Israelites, Egyptians, Germans, and Chinese bringing into existence new things and through mathematics solving some critical issues facing us in our world today. Computing has really evolved, and it keeps growing every day, from drones, to mini computers, to robots (does not necessarily have to be the metallic walking big robot we see in fictional movies, our washing machine is a robot, just a robot without commonsense), chips that can even be inside of man, computer vision, cryptography, encryption, face recognition, and so on, one could simply say artificial intelligence is the FUTURE, designs of expert systems cannot be overlooked as mathematics again is a basic for all expert system engine designs.

In other to describe the mutual impact of computing science and mathematics to each other, and their comparative roles, I am looking forward to the future, to the time where computing science would be a bit more mature and sure of itself, hopefully computing science and mathematics would one day exist as respected discipline, serving cognate but different roles in a person's education. According to George Forsythe, "the most valuable acquisition in scientific or technical education, are the general-purpose mental tools which remain serviceable for a lifetime. I rate natural language and mathematics as the most important of this tools, and computer science as the third".

Like mathematics, computing science will be seen as a subject which is considered rudimentary to general education. Like mathematics and other science, computing science will continue to be divided into two vaguely areas, which might be called "theoretical" and "applied". Like mathematics, computing science will be somewhat different from other sciences, in that it deals with man-made laws which can be proved, instead of natural laws which are never known with certainty. Thus mathematics and computing science will be like each other in many ways, the difference will be in the subject matter and approach, that is mathematics dealing with theorems, infinite process, and static relationships; and computing science on the other hand dealing with algorithms, finitary constructions, dynamic relationship; which are more or less mathematically inclined.

Many computing scientist have been studying mathematics, but in the real sense, many more mathematician have been doing a lot of computing science in disguise. Interestingly, numerous mathematical theorems which are really particular about algorithms, are typically formulated in mathematical terms that are more complex and less natural than the equivalent formulation that recent computing scientist would use. Before we go further, we should know a brief history of computer science. Here is a pictorial history of computer science (World science festival).

II. EDUCATIONAL SIDE EFFECT

A good computing scientist should know how to construct, manipulate, analyze and construct algorithms, this would be a general-purpose conceptual mental tool which will be a definite aid for understanding other subjects, and it also prepares one for more than just writing a good computer program. It is often being said that a person does not really understand something until he can teach someone else, meanwhile in the real sense a person does not really understand something until he can teach it to a computer in terms of algorithms. Computing makes this compulsory for all computing science as algorithms is a product of mathematics. The ability to express things as an algorithm leads to a much deeper understanding than trying to comprehend it the traditional way.

I believe that a properly trained computing scientist should learn something unique and spectacular that would implicitly help cope with other subjects, as we know computing science is linked to so many other subjects. On the other hand the present day undergraduate are not fulfilling this goals, at least from experience, many graduate students with undergraduate degree in computer science have been more narrowly educated, in fact graduate student with undergraduate degree in mathematics, physics or engineering perform better than ones with an undergraduate degree in computing science. Computing scientist of course-works however should thrive to solve this deficiency, which I believe is as a result of an over-emphasis on computer programming languages rather than algorithms, forgetting the fact that these computer programs developer did not just come about the programs without some algorithms and mathematics.

Nigerians on the other hand should introduce more mathematics and logical reasoning as a basic for computing sciences, pointing on the fact that no one actually wants to study a course for the passion but for the capital studying the course would fetch them in the labor market, so many people believe information technology is the future, so they rush into it like it is all about using a computer and not understanding its component. Same reason why we have very few student wanting to go into academics, as lecturing is seen by most youths as a lame job.

III. RELATIONSHIPS BETWEEN THESE COURSES

Computing science has been affecting mathematics in many ways such as computing as the name implies where mathematics is applied when hand computation seems very difficult. Secondly, there are vivid connections between computing science and mathematics in the areas of logic, algorithms, number theories and numerical analysis. Looking at some contemporary computing science area of specialization that is growing on a daily basis: we consider some of these courses and the importance of mathematics in their individual studies: Cryptography (take for instance the RSA algorithm which was developed by Ron Rivest, Adi Shamir, and Leonard Adleman. The algorithm was first publicly described in August 1977 and a patent was filed in December 1977; as the patent was filed after the discovery was made public, the algorithm received a patent in the US only in 1983. The RSA algorithm is the first example of an algorithm for public key cryptography. It also solves the problem of authentication with public key systems. Much like Diffie-Hellman, RSA draws on modular arithmetic. Its security mainly relies upon the difficulty of factorizing a large number which is a product of two large primes. The two primes are part of the private key, whereas the single large number, the product of these two smaller ones is part of the public key. Multiplying two prime numbers together is computationally like a one way process. It is easy to calculate: 757 x 977 = ? But given the product it's comparatively very difficult to factorize it to recover the two primes: 739 589 = ? x ? In practice the primes used by RSA are several hundred digits long, so methods to find such large primes are needed; fortunately such methods exist. Since the security relies on it being difficult to factorize large numbers we are fortunate (or unfortunate) there is no method to quickly factorize numbers). Database engine (the mathematical aspect of the development of the database engine is the understanding of matrices and linear algebra. The transpose, inverse, rows, columns and its manipulation in the design of an algorithm for databases engine).

Amongst others are artificial intelligence, and almost if not all areas of computing science, another impact of computer science has been an increase emphasis on constructions in all branches of mathematics. Replacing existence proofs by algorithms which construct mathematical objects has often led to improvements in an abstract theory. Another way in which algorithmic approach affect mathematical theories is in the construction of one-to-one correspondences, there often has been indirect proofs that certain type of mathematical objects are equinumerous (In mathematics, two sets A and B are equinumerous if there exists a one-to-one correspondence (a bijection) between them, i.e. if there exists a function from A to B such that for every element y of B there is exactly one element x of A with f(x) = y - Wikipedia); then a direct correspondence of the one-to-one shows that even more than these are true.

Discrete mathematics (is not the name of a branch of mathematics, like number theory, algebra, calculus, etc. Rather, it's a description of a set of branches of math that all have in common the feature that they are "discrete" rather than "continuous" - Google) especially combinatorial analysis has been given an added boost by the rise of computer science, as a matter of fact, to every other field that discrete mathematics is being applied. But actually the most important impact of computing science on mathematics is somewhat different from the above, I agree with Dr. Donald Knuth, who said that the most significant thing is the study of algorithm themselves, which has opened up a fertile vein for interesting new mathematical problems and provides a breath of life for many areas of mathematics which have been suffering from lack of new ideas.

Discrete mathematics is the math that is used extensively in computer science. It's used primarily to determine the running time of an algorithm. The math involved here is a combination of combinatorics, number theory, and algebra. There are also algorithm design procedures, such as dynamic programming, that use mathematics. Dynamic programming for example uses extensive number theory. Cryptography, a branch of computer security, is heavily reliant on primes and number theory as well. Depending on the field an individual goes into, he or she might use different fields of mathematics. As far as programming goes, it's not required. There are several programmers who do not use mathematics at all. However, people who do more theoretical work do use mathematics extensively, and some algorithmic programmers also use mathematics a lot

Charles Babbage, one of the “fathers” of computing predicted this in 1864: “As soon as an analytical-engine (general purpose computer) exists, it will necessarily guide the future course of science. Whenever any result is sought by its aid, the question will then arise – By what course of calculation can the result be arrived by the computer at the shortest time?” and again George Forsythe in 1958 said and I quote: “The use of practically any computing technique itself, raises a number of mathematical problems. There’s thus a very considerable impact of computation on mathematics itself, and this may be expected to influence mathematical research to an increasing degree.” Many intriguing mathematical problems arise when we try to analyze an algorithm quantitatively, to see how fast it will run on a computer. And one of the first mathematical theories to be inspired by computing science is the theory of languages, which now includes many beautiful results. Conversely, mathematics has a profound influence on computing science

that nearly all branches of mathematics has somewhat of computing science in them and how these could possibly be applied scientifically (computational).
In addition, a workshop on "New Connections between Mathematics and Computer Science" was held at the Isaac Newton Institute for Mathematical Sciences in Cambridge, England from the 20-24, November 1995. The workshop was supported by the Engineering and Physical Science Research Council of the United Kingdom, the London Mathematical Society and Hewlett-Packard's Basic Research Institute in the Mathematical Sciences. And the following topics were explained:

- Differential geometry of algorithms
- Developments in complexity theory
- Algebraic topology and distributed computation
- Dynamics of proof and computation
- Geometry of images and computer vision.

Surprisingly we come across most if not all of these in computing science. Mathematics appears in practically all areas of computer science. Even in software engineering, if you don't understand basic discrete mathematics, you will be very behind in your understanding of algorithms and data structures. Before computer science even existed as a field, algorithms were still developed by mathematicians. In computer graphics, you couldn't even start without a solid knowledge of linear algebra. For artificial intelligence, you'll want to be proficient in calculus and linear algebra, as well as several other mathematical fields depending on where you are headed. For theoretical computer science, you will want to be well versed in several topics if not all of undergraduate mathematics, particularly in probability.

A. DETAILED EXAMPLE

The Imitation Game. Alan Turing and Alonzo Church both independently came up with a resounding NO to the Entscheidung's problem. Turing did this in his seminal paper "On Computable Numbers With An Application To TheEntscheidungs problem": (Page on virginia.edu). He constructed the Turing machine, a supremely important theoretical general purpose machine, to reason about computability. The Turing machine remains actively in use today, and you will hopefully learn about them in a Theory of Computation course. They basically are just like the computer or phone we currently use, just a lot simpler. Alonzo Church also arrived at the No conclusion for the Entscheidungs problem, but he used a little something called Lambda calculus to define "effective computability". It should be noted that the lambda calculus is essentially a programming language and is the foundation for modern Functional programming.
What should be taken away about the lambda calculus and Turing machines is that they are both models of computation (and they happen to be equivalent!) The idea of computation was born from a question about mathematical logic. This is the founding relationship between computer science and mathematics. As you probably know, courtesy of The Imitation Game, physical machinery was central in WWII for breaking the Enigma. After the war, computer engineering also took off with the goal of a general-purpose computer like the Turing machine. The ENIAC is one such example which finished directly after the war. As hardware progressed with printed circuits, von Neumann architecture, etc., programming and programming languages followed through. These are journeys of their own and deserve more treatment in perhaps a different question. Things progressed on the theory side as well. Algorithms analysis blossomed and the field now known as Theoretical Computer Science also continued, soon travelling into ideas of Computational complexity theory in the late 50s - 60s.
In conclusion, more discrete mathematics courses should be introduced at undergraduate and possibly graduate level, because computing science is exceptional in the applications of these mathematical theories. Computing Science uses every piece of mathematics it finds fitting for a given purpose, being it algebra, calculus, numerical methods, statistics or whatever else. I would not say that Computing Science is a subset of mathematics, nor its descendant, but rather an angle of view, or filter through which the achievements of mathematics are selected for computational solutions. The Computing Science on the other side extends and enforces the mathematics by helping with different demanding tasks, which can be done by a computer. Mathematics is still a foundation for Computing Science, in other words if computing Science is on the throne, Mathematics would be the seat or the pillar or the crown as the case may be.

BIIOGRAPHY

1. Forsythe, G.E (1959). The Role of Numerical Analysis in an Undergraduate Program, this MONTHLY, 66, 651-662

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 10, October 2016

2. George Forsythe and the Development of Computer Science, Comm. ACM, 15 (1972) 721-726
3. Knuth, D.E (1965). A Class of Projective Planes. Trans, Amer, Math. Soc., 115, 541-549
4. Knuth, D.E (1965). Computer Science Department, Stanford University, California, 323-328
5. Lax, P.D (1970). The Impact of Computers on Mathematics, chapter 10 of Computers and their Role in the Physical Sciences, ed., by S. Fembach and A. Taub, Gordon and Breach, New York, 219-226
6. Peter Wegner, Three Components Cultures, Advances in Computers
7. Stallings, Computer Organization and Architecture, 8th ed., Generations of Computers, 17-59.