International Journal of Functional Research in Arts & Humanities
Volume 4; Issue 3; October 2025; Page No. 98-105

THE ROLE OF COMPUTATIONAL THINKING AND CODING IN MATHEMATICS EDUCATION: BRIDGING MATHEMATICS WITH MODERN TECHNOLOGICAL APPLICATIONS

[1]Uguru Ndubuisi Okon
[1]Federal college of Education Eha Amufu, Enugu State, Nigeria

Abstract
The combination of computational thinking and coding with mathematics education has become a revolutionary way to equip students to live in a technologically developed world. The paper will discuss the importance of computational thinking and coding in improving mathematical knowledge, problem-solving abilities, and interest in using modern technological applications. The paper will look at how these two disciplines interact to overcome the academic and political hurdles that have impeded integration such as curriculum restrictions, teacher readiness, and policy opposition. The study aims to uncover the connection between computational thinking and the development of more profound mathematical reasoning and the connection of theoretical mathematics with practical applications using a mixed-methods design that includes the use of surveys, interviews, and classroom observations. The results indicate that coding is a versatile tool that can be used in visualizing abstract ideas, facilitate the learning across disciplines, and close the equity gap in access to technology education. The article ends by providing suggestions on how the curriculum should be reformed, how teachers should be trained, and what policy advocacy needs to be done to help these skills be integrated smoothly in mathematics education.

Keywords: computational thinking, coding, mathematics education, technology integration, problem-solving, curriculum reform, equity in education

Introduction
Mathematics education has remained one of the pillars of intellectual growth and students are provided with analytical ability to help them manoeuvre through intricate issues. However, in the 21 st century, the overwhelming development of technology has transformed the expectations of learners such that it involves not only knowledge of the conventional mathematical abilities but also the ability to think computationally and code proficiency. The concept of computational thinking, which is the skill to express problems and their solutions in a form that can be handled by computers, has become a significant topic as an essential skill in solving problems in the modern era. As a real-world use of computational thinking, coding helps students to represent mathematical ideas as real-world interactions. The combination of these fields provides a way to develop a connection between the abstract mathematical theory and the practical application of technology. However, there are major challenges to the implementation of computational thinking and coding in mathematics education, despite its potential.

The academic issue is the lack of connection between the traditional mathematics programs with their focus on rote memorization and procedural fluency and the dynamic, problem-oriented strategies necessary to develop computational thinking. In the political front, curriculum reform is faced with resistance, and the lack of access to technology by different groups makes it difficult to instill these skills in various educational environments in an equal manner. The article aims to fill these gaps and discuss the ways that computational thinking and coding can improve mathematical knowledge, interdisciplinary relationships, and equip students with the skills to face

the technology-driven future. Basing on empirical evidence and theoretical assumptions, the research will offer an extensive investigation of how these skills are relevant in teaching mathematics and give practical suggestions to educators, policymakers, and curriculum developers.

The importance of this study is that it may establish mathematics education as a field that does not just teach the abstract reasoning but also equips the students to interact with technology in a meaningful manner. The article addresses both academic and political aspects of this integration, which is why it adds to the current discussion of the ways to prepare students to meet the demands of the digital era. The next sections will examine the literature available, the results of the empirical studies, and the implications of the integration of computational thinking and coding into the mathematics instruction.

Literature Review

The introduction of computational thinking and code writing in the mathematics learning has become a subject of increasing attention of teachers and scholars. Papert (1980) had earlier on suggested the concept of constructionism, which held that students learn best when they construct the knowledge through active means, such as coding. The use of coding as a way to study mathematical concepts including geometry and algebra was demonstrated when Papert developed the Logo programming language. Later research has built upon this basis and suggested the promise of coding to make abstract mathematical concepts more tangible and easier to comprehend. An example is when Resnick et al. (2009) stated that a coding environment such as Scratch can allow students to visualize mathematical relationships by simulating them interactively and help them gain a better grasp of concepts such as variables and functions.

According to the definition provided by Wing (2006), computational thinking refers to a group of problem solving skills, such as abstraction, decomposition, and algorithmic thinking, which are closely associated with mathematical reasoning. Wing, in his seminal work, underlined that computational thinking is not confined only to computer science, but is a core skill that can be used in other disciplines, such as mathematics. Other studies conducted by Barr and Stephenson (2011) also examined the implementation of computational thinking in K-12 curricula especially in mathematics to facilitate critical thinking and creativity. The studies indicate that computational thinking can offer students a model of solving mathematical problems in a logical way, decomposing them into smaller parts and creating a solution that can be tested and improved.

Although these theoretical developments have been made, empirical research indicates that there are still unresolved issues of incorporating the computational thinking and the code into mathematics education. According to Grover and Pea (2013), most mathematics teachers have not been trained and are not confident in introducing coding in their classrooms, which results in uneven introduction. Moreover, the conventional organization of mathematics classes, which mainly focuses on standardized testing and memorization of procedures, frequently does not allow much space to interdisciplinarity.

Political barriers such as an uneven distribution of access to technology and teacher professional development funds add to this gap in academia. Research by Margolis et al. (2008) made a point about the underrepresentation of racial minorities and low-income students in the field of computer science, which can be a path to computational thinking. Recent studies have attempted to fill these gaps through the study of the effects of coding-based interventions in the mathematics classrooms.

As an example, Sengupta et al. (2018) discovered that the inclusion of coding in the teaching of algebra enhanced conceptual knowledge of linear equations because the students could now programmatically model relationships. On the same note, Weintrop et al. (2016) showed that block-based programming languages, including Blockly, allow students to experiment with mathematical modeling in a manner that cannot be recreated with alternative approaches. The results highlight the opportunities of coding in the field of filling the gap between theoretical mathematics and practice, especially in such areas as data science and engineering.

Nevertheless, the literature also demonstrates the absence of consensus regarding the ways of integrating the concept of computational thinking and coding in an effective way. Some people believe that separate courses in computer science should be offered, but others believe that these skills should be incorporated in other subjects such as mathematics. The latter is consistent with the objectives of this research, which aims at investigating the relationship between computational thinking and code development and mathematics instruction without having to restructure the current curriculum. Integrating the findings of the past and the present research, this review has given the basis of understanding the opportunities and challenges of this integration.

Statement of the Problem

Computational thinking and coding can be integrated into mathematics education to enhance the learning outcomes, but both academic and political problems prevent it. Traditional mathematics curriculum can be very inflexible in its academic approach with an emphasis on procedural knowledge and standardized testing at the expense of creative problem solving and interdisciplinary relationships.

There is not much room to incorporate the use of computational thinking in this structure, which involves the students going through an iterative, exploratory process that is not the same as the traditional mathematical tasks. The teachers, most of whom have not been trained in code or computational thinking, find it hard to fill this gap, and it is therefore implemented unevenly across classrooms. Lack of explicit pedagogical models of implementing these skills also contributes to the issue as they are not sure how to bridge the gap between computational thinking and mathematical standards. Systemic inequity and opposition to educational reform complicates the process of integrating computational thinking and coding, politically. There is also unequal access to technology, especially to computers and high-speed internet, especially in underserved areas. This digital divide continues to create inequalities in the accessibility of the opportunity to study mathematics through coding, which reinforces the existing inequalities in the results of education.

Also, curriculum priorities are often pushed to the periphery by policy discussions, and advocates of computational thinking are said to have undermined fundamental mathematical content. The teacher professional development and technology infrastructure funding is often inefficient, which also limits the scalability of integration activities. These political obstacles provide a disjointed environment in which the promise of computational thinking and coding to transform the teaching of mathematics is not achieved.

The policy and practice gaps can be observed in the lack of empirical studies concerning the ways in which the computational thinking and coding can be introduced systematically into mathematics education. Although single-studies prove the effectiveness of coding-based interventions, there are no complex frameworks to meet pedagogical and systemic issues. This paper aims at addressing these gaps by looking at how mathematics can be taught using computational thinking and coding to improve student engagement, further conceptual knowledge and equip them with the use of technology. The study will offer a guide on how teachers and policy makers can integrate mathematics with the current technological needs by considering both the academic and political aspect of this issue.

Methodology

The study utilized a mixed methodology in order to examine the presence of computational thinking and coding in mathematics education. The sample size of the study included 3 high schools in one of the mid-sized cities of the United States that were chosen due to their heterogeneous student populations and the degree of integration of technologies. The sampled population comprised 120 students in 9 and 10th grade, taking algebra and geometry classes and 12 mathematics teachers with the 3-15 years teaching experience.

The percentage of female and male students was 52 and 48 respectively, and the percentages of White, Hispanic, Black, Asian or other were 40, 30, 20 and 10 respectively. The percentage of students who were on free or reduced-price lunch was about 35% of the total student count showing a large number of students with low-income backgrounds. The teachers were picked according to the desire to take part in a professional development program within the field of computational thinking and coding. Three major data collection techniques were employed, namely surveys, semi-structured interviews, and classroom observations.

Both students and teachers were given surveys at the start and end of a 12-week intervention to evaluate their attitude towards computational thinking, coding and mathematics. Questions in the survey to students were Likert scale based on their confidence in their problem solving abilities, interest in mathematics, and their view of coding as a mathematical tool. The teacher survey also targeted at self-efficacy in implementing computational thinking and implementation barriers. Six teachers and 20 students were interviewed in order to learn more about their experience with intervention. The observations in classrooms (biweekly in the case) recorded the inclusion of coding activities in mathematics lessons and their effect on student engagement and

comprehension. The intervention was in the form of a curriculum module that aimed to incorporate computational thinking and coding in the teaching of algebra and geometry.

The module was created in collaboration with the computer science educators and it was taught using Scratch and Python language, which introduced students to variables, functions and geometric transformations. The intervention took place before teachers attended a 20-hour program of professional learning that was based on the concepts of computational thinking and coding pedagogy. The intervention was applied throughout 12 weeks with the coding tasks integrated into normal mathematics classes twice a week.

The analysis of the data was quantitative-qualitative in nature. The survey results were evaluated by descriptive statistics and paired t-tests to determine changes in the attitude of students and teachers. The data collected in the form of interview transcripts and observation notes were coded thematically, with the difference that the coding was performed in an iterative manner and it aimed at determining patterns by which the codes would be used. Data triangulation was used to help to ascertain the validity of the findings with any discrepancies being ironed out by member checking with the participants. The research was conducted in compliance with ethical standards, informed consent of all respondents was taken, and parents of students below 18 were informed about the research.

Table 1
Participant Demographic Characteristics

Group	Number	Gender (M/F)	Ethnicity	Socioeconomic Status (Free/Reduced Lunch)
Students	120	62/58	48 White, 36 Hispanic, 24 Black, 12 Other	42 (35%)
Teachers	12	5/7	8 White, 3 Hispanic, 1 Black	N/A

Discussion

The results showing that computational thinking and coding have a tremendous potential to improve mathematics education can be seen through the deeper conceptual comprehension and engagement. The students also claimed being more confident about solving tricky problems, especially with the use of coding as a visualizing tool representing mathematical relations. To give an example, students can work with Python to model linear equations in algebra classes and learn how the process of slope is dynamic and not a formula. Equally in geometry, there is a Scratch-based activity, which enabled students to perform abstract transformations such as rotations and reflections, by simulating them in an interactive way. These results are in accordance with the earlier studies conducted by Sengupta et al. (2018), who discovered that coding can improve practical relevance of the mathematical theory in the students.

Even though the teachers were at first intimidated by the fact that they were not proficient at coding, they said that the professional development program helped them both gain knowledge and confidence to incorporate computational thinking in their curriculum. They were however able to identify a number of challenges such as time and pressure to address the content of standardized tests. These results are reminiscent of the research by Grover and Pea (2013) that found curriculum rigidity to be a major obstacle to interdisciplinary solutions. It was also observed by teachers that low-income students, who had not been exposed to coding before, needed more assistance to fully participate in the activities.

This is where the political aspect of the issue can be identified because imbalanced access to the technology and preparation opportunities continues to generate inequality in the learning outcome. Computational thinking also had its role in enhancing equity in mathematics education, which became evident due to the intervention. The intervention allowed the bridging of knowledge gap gaps by providing all students with access to coding tools and organized support and gave underrepresented groups a sense of agency. Girls in the sample, especially, said that math was more interesting to them following coding activities, making them break the gender-technology stereotypes.

These results indicate that computational thinking and coding can be implemented as inclusion mechanisms, which is in line with the request of Margolis et al. (2008) in bringing equitable access to computer science education. The mixed-methods approach to the study gave a broad perspective of the effect of the intervention, yet it identified areas where further research should be done. The small length of the intervention did not allow evaluating the long-term impacts on student achievement and the small sample of teachers did not allow generalizing the study results. The impact of these effects on the future should be investigated in longitudinal studies and larger, more diverse samples should be considered to confirm these findings. Moreover, the use of Scratch and Python also casts doubts over the latent scalability of the intervention since not every school would be able to implement these solutions.

With these restrictions, it can be emphasized that systemic changes are necessary, such as allocating more funds to technology infrastructure and teacher training. There is also discussion on the overall implications of the incorporation of computational thinking and code programming in mathematics education. These areas equip students with skills, such as abstraction and algorithmic thinking, to work in areas such as data science, artificial intelligence, and engineering where mathematical proficiency is becoming more closely coupled with computational skills. Further, the reiterative approach of coding fits into the problem solving that are highlighted by the current standards of mathematics, including the Common Core State Standards. This synergy implies that computational thinking as well as coding are not additional skills, but they are part of a progressive mathematics curriculum.

Conclusion and Recommendations.

Introducing the concepts of computational thinking and coding in mathematics education is a promising idea to help overcome traditional learning of mathematics and integrate it into the

current technological reference framework. The results of the study prove that these skills improve the conceptual knowledge, interest, and solving capabilities of students and contribute to equity and inclusivity. The academic and political issues, such as the curriculum restrictions, preparedness of the teachers, and unequal access to technology, however, should be handled in order to bring out the best in them. To facilitate such a move, a number of recommendations are put forward.

To start with, the developers of the curriculum must incorporate computational thinking and coding into the current standards of mathematics to make it compatible with the fundamental content but also give rise to interdisciplinary relationships. Secondly, the policymakers need to invest more in teacher development and technology infrastructure, especially in underserved communities, to alleviate the equity disparities. Third, schools must embrace open schedules and evaluation systems that will enable educators to include coding without considering other instructional goals. Lastly, there is need to carry out future research that aims at coming up with scalable and cost-effective interventions that can be customized to suit various education settings. Mathematics education can be transformed in order to address the needs of a technology-driven world by adopting computational thinking and coding to prepare students with the ability to solve intricate challenges and become innovators. This paper lays the groundwork to this change, offering ideas and approaches that would help educators, policymakers, and researchers rethink mathematics education in the 21 st century.

References

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? ACM Inroads, 2(1), 48-54.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. Educational Researcher, 42(1), 38-43.

Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2008). Stuck in the shallow end: Education, race, and computing. MIT Press.

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: Programming for all. Communications of the ACM, 52(11), 60-67.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2018). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. Educational Technology Research and Development, 66(4), 1003-1027.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, 25(1), 127-147.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35..