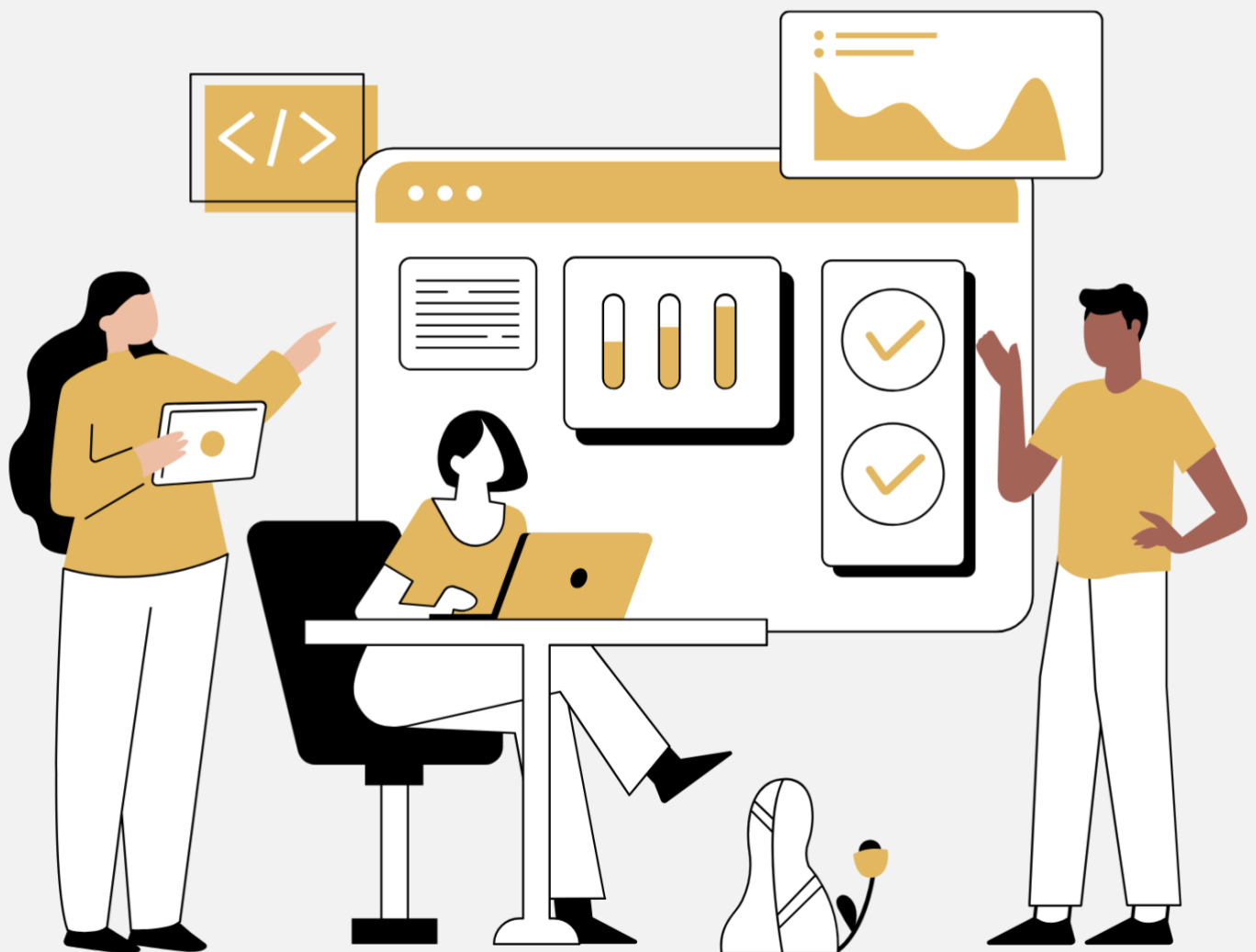


# **CODING, COMPUTATIONAL MODELLING, & EQUITY IN MATHEMATICS EDUCATION**

2 0 2 3  
April 26, 27-29  
St. Catharines,  
ON, Canada

## **2023 Symposium Proceedings**



[https://cpmath.ca/  
ccmeme2023/](https://cpmath.ca/ccmeme2023/)

SSHRC  CRSH  
Social Sciences and Humanities Research Council  
Conseil de recherches en sciences humaines

 Callysto

  
FIELDS

  
Brock  
University

  
Western  
UNIVERSITY - CANADA

CCMEME 2023 PROCEEDINGS

**CODING, COMPUTATIONAL MODELLING, AND EQUITY IN MATHEMATICS  
EDUCATION SYMPOSIUM**

April 26, 27-29, 2023

St. Catharines, Ontario

Canada

**Citation:** Buteau, C., Namukasa, I. & Sardella, J., Eds (2024). *Proceedings of the Coding, Computational Modelling, and Equity in Mathematics Education Symposium*. St. Catharines (Canada), April 2023.

## Foreword

Chantal Buteau<sup>1</sup> & Immaculate Namukasa<sup>2</sup>

<sup>1</sup>Brock University, <sup>2</sup>Western University

This Symposium + PD-Day was part of two legacies.

The [first legacy](#) is one of symposia organized at Western University. The first of this symposium, way back in 2001 and funded by the Fields Institute for Research in Mathematical Sciences, focussed on online learning. Lead organizers George Gadanidis and Bill Higginson hosted this event that included as keynote speakers, Seymour Papert and Robin Kay.

The [second legacy](#) is one of a two-decade history of engagement of teaching to use programming in project-based mathematics courses at Brock university. Computational modelling has been integrated in undergraduate mathematics courses in Year I, II, III for mathematics majors and future mathematics teachers, in the Faculty of Mathematics and Science. These MICA courses started in 2001. MICA stands for *Mathematics Integrated with Computers and Applications*.

The Brock research team came together with the Western research team, starting in 2017, to co-organize symposia, forums, webinars, seminar series and communities of Practice (supported by Federal Government, Provincial government and Fields funding). The [2015](#) and [2017](#) symposia, as well as the [2020](#) online seminar series, on *Coding and Computational Thinking (CT) in Mathematics Education* came at a crucial time when provincial, national and international interests in these topics spurred a movement of curricular, pedagogical and assessment revisions to include the development of these skills and ways of thinking among students starting in primary school.

This Symposium + PD Day was an opportunity to share, discuss and build partnerships around approaches, perspectives and experiences that advance our understanding on diverse important issues of practical and theoretical nature concerning this growing area.

*Chantal Buteau*

Chantal Buteau  
Brock University

*Immaculate Kizito Namukasa*

Immaculate Kizito Namukasa  
Western University

## Committees

### **Symposium Scientific Committee**

Chantal Buteau (Brock University) – Co-Chair  
Immaculate Namukasa (Western University) – Co-Chair  
Laura Broley (Brock University)  
Ghislaine Gueudet (Université Paris-Saclay, France)  
Joyce Mgombelo (Brock University)  
Marisol Santacruz Rodríguez (Universidad del Valle, Colombia)  
Ana Isabel Sacristán (Cinvestav, Mexico)  
Ricardo Scucuglia (UNESP, Brazil)

### **PD Session Committee**

Steven Khan (Brock University) – Co-Chair  
Marja Bertrand (Western University) – Co-Chair  
Anjali Khirwadkar (Brock University)  
Dave Potts (Brock University)  
Derek Tangredi (Western University)  
Shannon Welbourn (Brock University)

### **Local Organization Committee**

Laura Broley (Brock University) – (former Chair until June 2022)  
Dorothy Levay (Brock University) – Co-Chair  
Chantal Buteau (Brock University) – Co-Chair  
Steven Khan (Brock University)  
Joyce Mgombelo (Brock University)  
Neil Marshall (Brock University)

### **Organizing Committee**

Laura Broley (Brock University)  
Chantal Buteau (Brock University)  
Zeynep Gecu-Parmaksiz (Ontario Tech University)  
Steven Khan (Brock University)  
Dorothy Levay (Brock University)  
Immaculate Namukasa (Western University)  
Beyza Sezer (Western University)

## Links to Recordings

Find below the links to the recordings from the Coding, Computational Modelling, and Equity in Mathematics 2023 Symposium, as well as a picture summary of the event.

*Symposium en Image:* <https://www.youtube.com/watch?v=7mUo5-y9hC4&feature=youtu.be>

*Opening Ceremonies:* <https://www.youtube.com/watch?v=io89D7D7DvA>

*diSessa Keynote & Davis Reaction:* <https://www.youtube.com/watch?v=6GYUgGuKf7c>

*Christian Keynote & Eglash Reaction:* <https://www.youtube.com/watch?v=BzVncPkRUqM>

*Working Group Reports:* <https://www.youtube.com/watch?v=1ToEfZEvtdM>

*Research Discussion Panel:* <https://www.youtube.com/watch?v=tH-wgZW0f0M>

*Practice Discussion Panel:* <https://www.youtube.com/watch?v=8YCMSvs010M&t=1s>

*Closing Ceremonies:* [https://www.youtube.com/watch?v=0zXf\\_U0aCzM](https://www.youtube.com/watch?v=0zXf_U0aCzM)

# Table of Contents

<a href="#">Foreword</a> .....	I
<a href="#">Committees</a> .....	II
<a href="#">Links to Recordings</a> .....	III
<b>Symposium Activity Report</b>	
<a href="#">Coding, Computational Modelling, and Equity in Mathematics Educations: P.D. Day &amp; Symposium Report</a> , <i>Wendy Ann Forbes &amp; Alexandria Middlemiss</i> .....	1
<b>Working Group Reports</b>	
<a href="#">Coding, Computational Modelling &amp; Equity in Mathematics Education: Working Group A Early Years/Elementary Report</a> , <i>Iain Brodie, Steven Khan, &amp; Sandy Youmans</i> .....	9
<a href="#">Coding, Computational Modelling &amp; Equity in Mathematics Education: Working Group B Secondary/University Report</a> , <i>France Caron, Steven Floyd, &amp; Miroslav Lovric</i> .....	19
<a href="#">Coding, Computational Modelling &amp; Equity in Mathematics Education: Working Group C Equity, Diversity, and Inclusivity Report</a> , <i>Diane Tepylo, Annie Savard, &amp; Ricardo Scucuglia</i> .....	33
<b>Keynotes &amp; Reactions</b>	
<a href="#">On the Nature of Literacies and Literacy Development</a> , <i>Andrea diSessa</i> .....	43
<a href="#">Some Thoughts on Andrea diSessa’s Five Powerful Ideas about Technology and Education</a> , <i>Brent Davis</i> .....	51
<a href="#">Coded Bias: Decoding Racism in AI Technologies</a> , <i>Gideon Christian</i> .....	56
<a href="#">Anti-Bias and Pro-Transformation: How to Merge Critique and Transformative Visions for Artificial Intelligence</a> , <i>Ron Eglash</i> .....	63
<b>Discussion Panels</b>	
<a href="#">How Do Computational Thinking and Mathematical Thinking Interact (in Terms of Knowledge, Ways of Thinking, and Competencies)?</a> , <i>Eirini Geraniou, Paul Drijvers, &amp; Elise Lockwood</i> .....	67

<a href="#"><u>Mathematics Education Incorporating Coding: Practical Challenges and Opportunities</u></a> , <i>Celia Hoyles, George Gadanidis, Oh Nam Kwon, Simon Modeste, &amp; Elena Prieto-Rodriguez</i> .....	73
---	----

**Other Contributions**

<a href="#"><u>Using Coding to Enhance Numeracy Teaching and Learning</u></a> , <i>Andrijana Burazin, Taras Gula, &amp; Miroslav Lovric</i> .....	84
---	----

<a href="#"><u>Coding and Computational Thinking: A Time that Came and Went but Will Always Be Ours</u></a> , <i>Elena Prieto-Rodriguez</i> .....	93
---	----

<a href="#"><u>Re-Designing Coding-Based Mathematics Tasks into Scaffolded and Project Formats: Two Pre-Service Teachers' Perspectives</u></a> , <i>Samantha Boerkamp &amp; Jessica Sardella, with Chantal Buteau</i> .....	98
---	----

<a href="#"><u>Integration of Coding and Computational Thinking in Compulsory Education: The Landscape in Canada - A Report</u></a> , <i>Emma Molinaro, with Chantal Buteau</i> .....	107
---	-----

**Post-Symposium Reflections**

<a href="#"><u>Computational Thinking: A Reflection on the Symposium by an Early Career Researcher in Undergraduate Mathematics Education</u></a> , <i>Francis Duah</i> .....	119
---	-----

<a href="#"><u>A Perspective of the Experience at CCMEME Symposium</u></a> , <i>Carolina Yumi Lemos Ferreira Gracioli</i> .....	121
---	-----

<a href="#"><u>A Reflection on the Coding, Computational Modelling, and Equity in Mathematics Education Symposium</u></a> , <i>Sheree Rodney</i> .....	123
--	-----

<a href="#"><u>Some Reflections Emerged from the Conference</u></a> , <i>Marie-Frédérick St-Cyr</i> .....	125
---	-----



## **Coding, Computational Modelling, and Equity in Mathematics Education**

### **P.D. Day & Symposium Report**

April 26, 27-29, 2023

Wendy Ann Forbes<sup>1</sup>, Alexandria Middlemiss<sup>2</sup>

<sup>1</sup>PhD Candidate, Brock University; <sup>2</sup>PhD Candidate, Western University

In mathematics classrooms across the globe, educators are increasingly integrating various forms of plugged-in and unplugged coding activities with the interest of promoting knowledge and skills related to computational thinking (CT). Plugged-in coding activities are taught via block-based and text-based computer programming software such as Scratch and Python, respectively. Unplugged coding activities typically involve screenless methods, including paper instructions, game-based learning, manipulatives, robotics etcetera. The various forms of practical applications in mathematics classrooms arise from increased research in CT, promulgated by societal changes. Certain societal changes are due to the rapid pace of technological developments, and these necessitate learner's computational participation in areas such as artificial intelligence, robotics, and coding. Notwithstanding, the concept of CT remains somewhat elusive, giving rise to ongoing concerns about equity in mathematics education and the extent to which K-12 teachers are prepared to implement coding-focused mathematics curricula, and artificial intelligence-focused mathematics curriculum.

The Professional Development (PD) day and Symposium on Coding, Computational Modelling, & Equity in Mathematics Education (CCMEME), hosted by Brock University on April 26, 27- 29, 2023, was an effort to address such emerging issues highlighted by research. The PD day and symposium involved workshops, working groups, plenary, panel discussions, and reactions. The organization of the event brought together two SSHRC-funded research teams hosted at Brock University (led by Dr. Chantal Buteau) and Western University (led by Dr. Immaculate Namukasa), who work on programming and computational participation across all levels of mathematics education (PreK-16).

The complimentary PD on April 26 was a precursor to the symposium, with pre-registration numbers over 100, and participation numbers over 60 in person participants including pre-service and in-service teachers, educators, and young and experienced researchers. The hands-on nature of the variety of plugged-in and unplugged coding workshops, facilitated by a range of educational partners and educators across Canada, offered practical suggestions of how teachers and mathematics educators of all levels may integrate coding and computational Modelling activities into CT-focused mathematics curricula. Workshop topics included, but were not limited to, Artificial Intelligence and Scratch, Integrating Coding in the Curriculum, Incorporating Data Visualization Exercises, Introduction to AI for Educators, and Introduction to Thinking Classrooms, and Robotics.

The symposium that followed on April 27- 29, 2023 registered 63 educators, scholars, graduate students and other stakeholders of diverse backgrounds in-person, with other participants attending the online sessions. Attendees, presenters and panelists spanned three working groups, representing the United States, Mexico, Brazil, France, Norway and Australia and provinces from across Canada. Additionally, virtual presenters and panelists represented France, the United Kingdom, the Netherlands, and South Korea. Of the three working groups, two focused on coding and computational Modelling in mathematics education (one group for elementary and early years, another group for secondary and post-secondary), and one group





focused on equity, diversity and inclusivity in coding and computational Modelling. Over the three days of the symposium, attendees spent several hours with their working groups sharing ideas, participating in activities and generating knowledge together.

The plenary sessions, panel discussion and the three working groups facilitated activities and discussions supporting the following themes of the symposium:

1. Meanings and interactions between “computational thinking” and “mathematical thinking.”
2. Integration of coding and computational Modelling in math curricula and classrooms at different levels of education.
3. Theoretical and practical understandings related to improving equity in integration of CCM in mathematics education.

On the first evening of the symposium, Dr. Andrea diSessa, Professor Emeritus in Education at UC Berkeley, amalgamated years of experience, research and knowledge in his keynote presentation titled “Five Powerful Ideas about Technology and Education.” Dr. diSessa synthesized 40 years of work in computers and education, presenting a concise list of five essential ideas crucial for progress in education. Given the extended developmental and uncertain trajectory of computers - a once in several centuries innovation - he emphasized the necessity of these ideas.

Firstly, Dr. diSessa asserted that beyond teaching coding, we should embrace a new literacy based on computational media, intended to augment rather than abandon the traditional literacy based on text. He argued that developing classic literature and the social infrastructure for computation literacy takes time, but it is necessary to avoid stagnation in the past or becoming solely reliant on contemporary technology. Furthermore, Dr. Di Sessa emphasized that there are certain myths that must be dispelled about computational media to be able to educate from a perspective that facilitates genuine appreciation for new literacies based on them. Prevalent myths include, for example, the idea that technology is becoming better and better, therefore, there is not much to be learned about it; the use of advanced technology is sufficient for learning (one-way literacy); and the focus of teaching students how to program is so that they will get a job (vocationalism). Moreover, adopting a polemical stance, Dr. diSessa contends that we need to reconsider the popular trend in research that considers computational thinking (CT) as a literacy as CT is blind to real-knowledge (such as physics and mathematics).

Secondly, he explored the concept of re-mediation, emphasizing that representations change the intellectual world - what we can think and know (material intelligence). Dr. diSessa framed this idea within the historical context of how algebra transformed the field of science. He then seamlessly transitioned to the next two big ideas. Thirdly, he emphasized that computational representations change the nature of engagement and activity structure. Fourthly, he pointed out that the representations that computers afford through, for example programming, provide open toolsets of genetic resources for learners to adapt to specific topics. Finally, Dr. diSessa summarized his fifth big idea by explaining the LaDDER model - Layered, Distributed, Development of Education Resources. Using this acronym, he advised that educational toolsets and learning activities should be developed in a close but appropriate relationship with the classroom to maximally empower both teachers and students. He stressed the importance of granting teachers agency, that is, providing them with resources and ideas so that they can solve their own problems.

The conversation was furthered by a deeply thoughtful and challenging reaction from Dr. Brent Davis of University of Calgary. Dr. Davis interpreted Dr. diSessa’s presentation as a

manifesto, promoting computational literacy to do mathematics and science differently. However, he questioned the definition of computational literacy and, due to this dearth in understanding exactly what computational literacy is, he also questioned Dr. diSessa’s activist perspective. Nevertheless, he challenged symposium participants to imagine their positions individually or collectively in this computational literacy discourse, whether as an observer describing or as an analyzer explaining or as an activist intervening. Davis also questioned the possibility of change in the near future, given the current state of institutions.

On the second day of the symposium, a discussion panel was held with a research focus, centering on the question: “How do computational thinking (CT) and mathematical thinking (MT) interact (in terms of knowledge, ways of thinking, and competencies)?” The panel, held virtually and livestreamed to attendees, was chaired by Dr. Ghislaine Gueudet (University Paris-Saclay, France). Panelists Dr. Paul Drijvers (Freudenthal Institute, Utrecht University, Netherlands), Dr. Eirini Geraniou (University College London, UK), and Dr. Elise Lockwood (Oregon State University, USA) brought their expertise together to relay knowledge, orchestrate an insightful discussion, and unearth the deep connections between CT and MT. As a precursor to the panel discussion, attendees answered three questions using an online app for responses, the Padlet, serving as an initial inventory of symposium attendees’ experiences and views related to the interactions between CT and MT (focused on personal, societal, and educational perspectives). The results indicated that attendees had productive experiences of the interaction of CT and MT, for example, using coding software with students or even in their own working groups over the course of the symposium. However, the poll responses also revealed that there were still questions to be explored. One attendee pointed out the need for a common definition of computational thinking and mathematical thinking, and the distinction between the two, in order to answer the question. Against that backdrop each panelist put forward their augment.

Dr. Drijvers explored the relation between CT and MT, pointing out that we may begin to think about how they relate by inspecting where respective skills overlap (e.g. abstraction, generalization, decomposition, algorithms, problem solving etc.). Drawing on the works of Papert (1980) and Lodi (2020), he discerned computational thinking in mathematics education as structural problem-solving. This involves using thinking processes and skills common to CT and MT to solve or outsource mathematical problems to an external agent, whether that be another human or a machine. Giving several options of relational possibilities, Dr. Drijvers gave concrete examples of activities and tasks, both plugged and unplugged, showing the interaction of CT and MT.

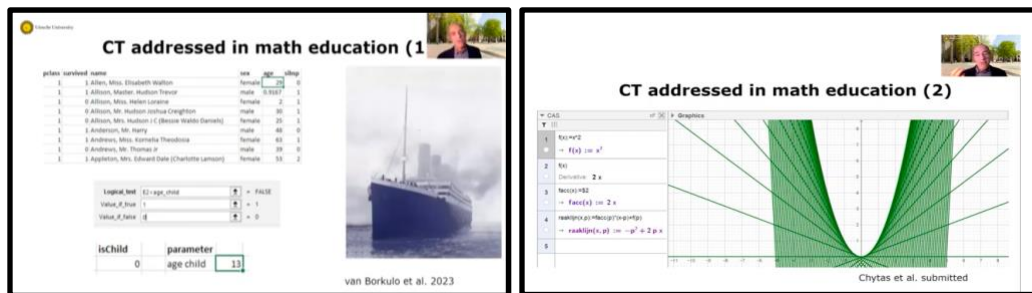


Figure 1. An illustration by Dr. Drijvers of plugged and unplugged activities.

In Dr. Drijvers’ first example, students were provided with a spreadsheet containing Titanic passenger’s data. Using CT and MT, students were tasked with employing specific variables to analyze the data by performing ‘data moves.’ The example shown involves counting the number of children aboard. In the second example, students were asked to input tangent lines

to Geogebra in order to create a specific graph as the program’s output. Dr. Drijvers concluded by highlighting that though CT and MT share common ground, further research is necessary to cultivate this shared territory.

Dr. Lockwood agreed that while CT and MT may have distinctive skills on their own, they also share a common ground. To illustrate her point, she highlighted her research in combinatorics (counting) problems, demonstrating that the existing limitations in helping students to connect counting processes to a set of outcomes is made easier by assisting them to develop simple programming tasks in Python. With the evidence that CT supports combinatory thinking, Dr. Lockwood was clear about her motivations to explore theoretically sound methods for researching how computational thinking can support what is known about mathematical thinking. Her closing statement prompted further exploration of how computational thinking and mathematical thinking support each other.

Lastly, Dr. Eirini Geraniou presented from her unique perspective as a researcher specializing in mathematics educators’ perceptions of CT and how it interacts with mathematics education, as well as advocacy for CT integration in mathematics education. She began, like Dr. Drijvers, by pointing to varied but interrelated versions of the definition of CT over the years, following up with the question prompted by an assertion in Sands et al.’s (2018) research that CT is synonymous with mathematics that uses digital tools. Dr. Geraniou asks: “Are CT and MT only linked when using digital technologies?” Through her research, she has identified the ways in which mathematical and digital competencies are interconnected, developing the construct of MDC - Mathematical Digital Competency - pointing to key identifiers that demonstrate MDC in students (Geraniou & Jankvist, 2019).

Concluding remarks linked the panel discussion to the overall themes of the symposium, emphasizing the need for systemic changes in the field to advance equity and access. Attendees were also invited to participate by responding to additional poll questions using Padlet toward the end of the panel, discussing two thought-provoking vignettes - the ‘Adventure Travel List Problem’ and ‘Math Ed in the Future.’ These discussions speculatively explored how artificial intelligence might be applied in mathematics education. This culminating activity transitioned into a poster session during the lunch break.

Poster presenters represented the scope of the working groups, highlighting research and relevant practices aligned with the symposium’s themes. In total, there were 19 posters: three focused on innovative practices, and 16 that were research-focused.

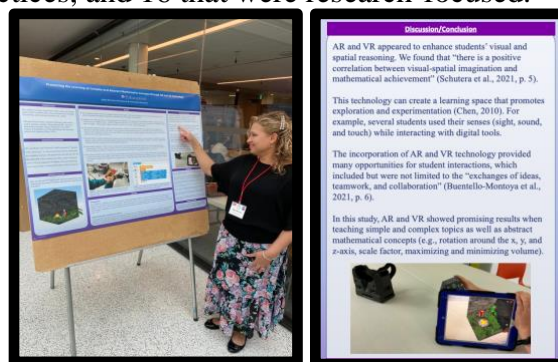


Figure 2. Marja Bertrand, a PhD Student at Western University, presents her poster Promoting the learning of complex and abstract mathematics concepts through AR and VR technology in the photo on the left. The Discussion/Conclusion section of her poster is displayed on the right.



On the evening of Day 2, the second keynote speaker, Dr. Gideon Christian from the Faculty of Law at the University of Calgary, delivered a presentation titled “Coded Bias: Decoding Racism in AI Technologies”. Departing from the abstraction of coding and computational Modelling and its many applications in everyday life, Dr. Christian focused on the potential unintended consequences of these technologies on regular people in society. Dr. Christian’s years of experience in law using critical race theory (CRT) to identify barriers and racism in new and emerging technologies, especially in artificial intelligence (AI), came across quite vividly as he explained how AI can perpetuate racism. He preambled his presentation with an overview of CRT, dispelling common misconceptions about it before explaining how AI, one of the most transformative human inventions, is now subtly undermining the great progress that has been made in the area of civil rights to uproot elements such as of anti-black racism in society. Highlighting that the intent of CRT is to uncover racism embedded in structural institutional systems including law and policies, Dr. Christian keyed in on two aspects of AI that are biased toward people of color, particularly black people: (1) the use of AI in recidivism risk assessment, and (2) the use of AI in facial recognition systems, especially their use in law enforcement and in criminal justice systems. He noted that while the use of AI technologies in these areas has been touted as major innovation, their use comes with major concerns about their potential to perpetuate barriers and discrimination against black people and people of color. Dr. Christian provided several examples illustrating how black people are disproportionately represented in issues of misidentity in law enforcement and other areas in comparison to other groups. He explained that AI perpetuates barriers and racism when biased data is used to generate algorithms to train AI systems, emphasizing the concept of “garbage in, garbage out” in relation to data. In conclusion, Dr. Christian suggested that, given the high inaccuracy pertaining to some racial groups, it might be necessary to pause the use of AI tools on groups that are disproportionately affected. His presentation illuminated the need for educators and stakeholders who are connected to the development of AI systems to consider their role in maintaining structural biases, discrimination and racism.

The reaction to Dr. Chistian’s keynote was done by Dr. Ron Eglash, professor in the School of Information at the University of Michigan with a secondary appointment in the Penny W. Stamps School of Art and Design. Drawing on his extensive experiences in culturally situated technological education and automation for generative justice, Dr. Eglash’s reaction was an enriching extension of the discussion. While he endorsed Dr. Chistian’s presentation, he emphasized the need to supplement an anti-bias framework with transformative justice and vision in computer science and mathematics education. Dr Eglash argued that if stakeholders solely focus on anti-bias in their educational work, this may lead to misunderstanding by students. He identified three challenges with exclusively projecting the negatives of computing: the belief that removing bias makes systems inherently fair; the perception that entering computing implies abandoning social justice issues; and the failure to challenge students to imaginatively develop new technologies for the purpose of transformative justice. Dr. Eglash, therefore, proposed adding aspects of ethno-mathematics and ethno-computing, heritage algorithms, and AI to critical tools may, from a transformative perspective, include different marginalized groups, thereby promoting equitable computing. To illustrate his point, he presented numerous examples of computational models developed with various ethnic groups including African-American and Indigenous communities.





Figure 3. An illustration by Dr. Ron Eglash of ethno-computing and heritage algorithms.

On the third and final day of the symposium, organizers collaborated with the Fields Math Ed. Forum to host a hybrid practice-focused panel discussion. The central question guiding the discussion was: “What are the challenges and opportunities of integrating coding in mathematics classrooms?” The panel was chaired by Dr. Dame Celia Hoyles (UCL Institute of Education, University College London, UK), with perspectives on the topics presented by Dr. George Gadanidis (Western University, Canada), Dr. Oh Nam Kwon (Seoul National University, South Korea), Dr. Simon Modeste (University of Montpellier, France), and Dr. Elena Prieto-Rodriguez (University of Newcastle, Australia).

Prof. Dame Hoyles opened the panel discussion by expressing her conviction, based on many years of experience, that despite challenges, there are potentials to integrate ideas of CT/ computational literacy, and programming into pedagogical practice to make a significant difference in mathematics learning. She emphasized that some of these potentials had already been mentioned in the symposium, such as coding making mathematics learning more dynamic and involved for students. Prof. Dame Hoyles further highlighted that coding is ubiquitous - a phenomenon all over the world - such that children are doing it at home, in their clubs, etc., with strides being made since the pandemic. As a segway to introducing the panelists, she pivoted to a discussion about the ways coding is being integrated into school curricula. Giving a brief historical background about the introduction of coding in mathematics education, referencing Seymour Papert’s work on the consequential disruption, Prof. Dame Hoyles invited the panelists to make their presentations.

The diversity of the research panel allowed for deep and enriching perspective-sharing across contexts, made possible again through a hybrid synchronous format. Firstly, Dr. Kwon outlined a brief historical overview of coding and AI curriculum in South Korea in elementary and secondary education, concluding with several examples of coding education implemented in Korean classrooms. Dr. Modeste then outlined how students engage with coding and programming using project-based approaches in schools in France. Attendees were presented with specific examples, including video recordings of students working through a problem in Scratch. Dr. Gadanidis began his presentation with an example of a Grade 3-4 scratch puzzle, wherein students worked with variables to create spirals (pictured). Importantly, he emphasized that in such activities, the role of the teacher is not to explain, but to facilitate students explaining amongst themselves as they engage in play and exploration.

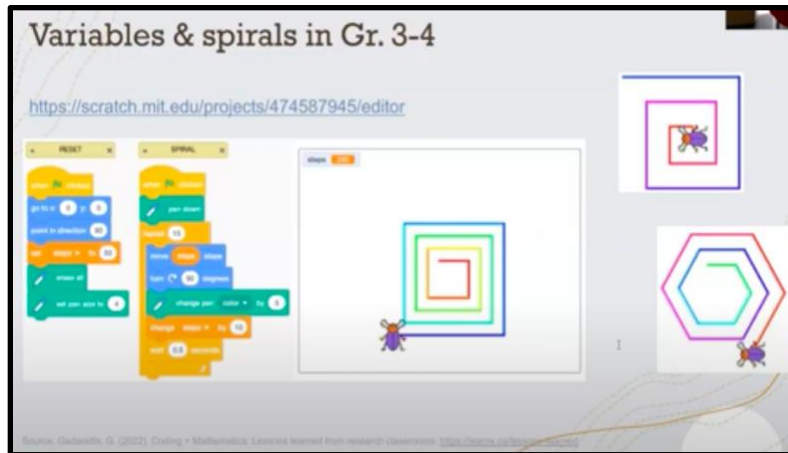


Figure 4. An illustration by Dr. George Gadanidis of a Grade 3-4 scratch puzzle.

Dr. Gadanidis proceeded to showcase more complex examples of programming using mathematics concepts as he transitioned to higher grades' curriculum and activities, using both Scratch and Python. After demonstrating potential practical uses, Dr. Gadanidis discussed how coding has become a new literacy as it is now mandatory in Ontario schools ( serving over 1.5 million students). Drawing on Andrea diSessa's work, Dr. Gadanidis argued that coding may revitalize mathematics education by means of transforming how and what is taught in classrooms.

Dr. Prieto-Rodriguez provided an overview of her collaboration with Dr. Dame Hoyles and Dr. Noss on the integration of coding and computational thinking into mathematics curriculum in New South Wales, Australia using ScratchMaths. The project aimed to explore elementary teachers' perception of facilitating students' learning processes to develop mathematical ideas through coding, and how those perceptions change after eight weeks of professional learning. Dr. Prieto-Rodriguez highlighted that the results revealed statistically significant positive outcomes, impacting the knowledge and attitudes of both teachers and students. Her research was motivated by challenges she observed that hindered the national mandate of making CT and information systems mandatory for all Australian students from Kindergarten to Year 8.

After an opportunity for questions, Dr. Richard Noss, Emeritus Professor of Mathematics Education at University College London Institute of Education in the United Kingdom, provided a brief reaction. He emphasized that the symposium is a tangible representation emerging from years of contribution from many researchers who have theorized about the effectiveness of the computer for learning. Dr. Noss noted that the examples from Dr. Gadanidis' presentation suggested that the focus has shifted to *when* is the computer good for learning instead of *is* the computer good for learning. In general, he noted that it appears that significant progress has been made in research concerning the integration of CT in mathematics education. Building from there, Dr. Hoyles asked the following: Given that Noss's observation is true, how are we going to ensure that the progress in research changes the culture in school mathematics? This prompt segwayed the panel discussion into a lively Q & A involving attendees and panelists. The discourse drew out a diverse range of philosophical, pedagogical, and practical approaches and perspectives.

On the final day of the symposium, each working group had the opportunity to present their work to all attendees. As anticipated, the themes that emerged from the working group discussions were in alignment with the larger themes of the symposium. The picture attached

shows a sample working group. Notably, Working Group A explored the notion that, for the elementary and early years, the focus of teaching should be building experiences (schema) within a sociocultural/societal framework. The aim of this is to provide students with future possibilities to think mathematically (proto-mathematics). According to the working group, coding enhances this experience; however, coding is not the mathematics itself.

In Working Group B, the focus was on sample coding and computational Modelling activities. Guiding questions were used to prompt participants to reflect on the activities. One salient point from this group is that computational Modelling is a way of bringing mathematicians, scientists, and computer scientists together - bringing mathematics into the mainstream of science - to address the complexities of today's most-pressing challenges.

Working Group C's activities were centered around developing a vision for Equity, Diversity and Inclusivity (EDI) using, as a lens, the 5 As (five affordances) of computational thinking (Agency, Access, Abstraction, Automation, and Audience) from Gadanidis' 2017 work. By the end of the three-day activities which included targeted small group engagement, they had developed an emerging framework for adapting coding activities to make them more inclusive. The framework includes three areas of considerations for reframing a task with respect to EDI principles: planning, implementing and assessing/building. At the planning stage, careful consideration is given to who the task is geared towards; at the implementation stage, the focus is on observing students to progressively create conditions for agency; at the assessing/building stage, the focus is on taking students from where they are to where they want to go.

Overall, the questions raised by the groups - pertaining to changes in tasks, tools for teaching, frameworks, classroom engagement, supporting pre-service and in-service teachers, and understanding mathematics in the context of CT - suggest that there is still much to be explored in research to elucidate the connection between CT and mathematics.

Finally, the summary of this report for the Professional Development (PD) day and CCMEME Symposium at Brock University reveals themes that suggest that the set objectives were met via participants' engagement in activities which includes coding workshops (with plugged-in and unplugged coding tasks), plenaries, panel discussions, a poster session, and working group discussions and reports.

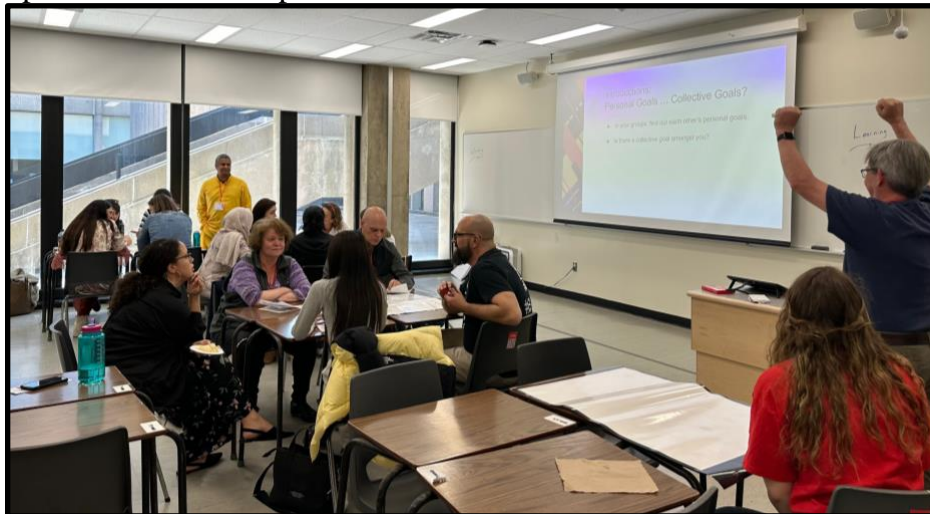


Figure 6. Working Group A leaders and members at the symposium.



## Coding, Computational Modelling & Equity in Mathematics Education Working Group A Early Years/Elementary Report

### *Working Group Leaders*

Iain Brodie<sup>1</sup>, Steven Khan<sup>2</sup>, & Sandy Youmans<sup>3</sup>

<sup>1</sup>Ontario Tech University; <sup>2</sup>Brock University; <sup>3</sup>Queen's University

### *Working Group Members*

Priscila Correa  
Brent Davis  
Jeanne Dobgenski  
Zeynep Gecu-Parmaksiz  
Anjali Khirwadkar  
Brian R. Lawler

Irina Lyublinskaya  
Kelly McKie  
Alexandria Middlemiss  
Siri Krogh Nordby  
Martine Rekstad

Marisol Santacruz-  
Rodriguez  
Pinar Sen  
Beyza Hatice Sezer  
Yimei Zhang



Figure 1. Working Group participants and co-leaders on last day.

### **Abstract**

Coding and computational Modelling are increasingly becoming curricular requirements across all age-ranges (Floyd, 2022). This working group (WG) explored this phenomenon by focusing on elementary and early years education with an emphasis on mathematics while acknowledging and respecting the interdisciplinary nature of the work of teachers during this dynamic and diverse period of child development. The working group's focus included:

- discussion of affordances and limitations of different metaphorical frames and descriptions for coding used in EEYME such as 'literacy' (diSessa, 2017), recipe, puzzle, playground (Bers, 2020), creative expression, and architecture (low floor, high ceilings, wide walls).
- developing and evaluating instructional materials and approaches for teaching coding and computational thinking in elementary and early years math classrooms (earlymath.ca)
- examining the impact of coding and computational thinking on students' math skills and attitudes
- exploring the use of technology and software tools to support coding and computational thinking in EEYME.
- investigating the ways in which coding and computational thinking can be used to enhance problem-solving and critical thinking skills in EEYME.





- sharing research findings, learning trajectories, and emerging best practices for teaching coding and computational thinking in EYME.
- critical discussion about under-articulated or less explicit goals such as computational participation (Kafai & Burke, 2017), productive computational disposition (Pérez, 2018) as well as opportunities for considerations of ethics, equity, and the development of empathy (Bers, 2022).

The WG was dedicated to examining how and why opportunities to code across the early years through elementary school provokes mathematics learning with a combination of hands-on activities and critical discussions. Participants had the opportunity to try out some unplugged and plugged tasks that have been tested in the mathematics classroom and explored the affordances of the Micro:bit (microbit.org) as a potentially equitable tangible coding platform.

## Introduction

Coding holds great promise for elementary and early years mathematics education because it fosters computational thinking, problem-solving, mathematical modelling, and creativity. By incorporating coding into mathematics education, we can deepen mathematical understanding and broaden instruction with a more playful and accessible approach (Gadanidis, 2015; Papert, 1990). However, most teachers at this level have not had formative and foundational experiences necessary to develop sufficient knowledge and confidence in coding and to use it to teach mathematics (Weber et al., 2022). Elementary teachers require opportunities to develop an understanding of coding in the curriculum and the knowledge needed to use it effectively with children, including enriching their learning of mathematics (Angeli et al., 2016). This working group provided an opportunity for members to share and build their coding capacity, while considering ways to develop coding abilities in elementary educators.

## Organizing Framework (YALL)

In preparation for the working group the leaders chose to adopt a framework previously developed and used by Lorraine Godden and Sandy Youmans originally called “Affirming, Yearning, and Learning”. The framework order was flipped to begin with Yearnings (What do we want to know/learn? Or why come to this working group? And followed by Affirming - What do we want to know? And what do we already know? The final part of the framework led into action - learning at large group and smaller sub-group levels (LL). The framework structured our engagements over the three days.

## Day 1

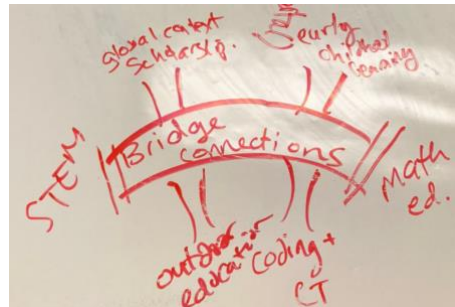
### Yearning (Motivations)

After introductions, table groups were asked to identify their personal goals for attending this working group and to begin to articulate potential shared collective goals. Following the idea of Simon Sinek (2009) the orienting question in the first session was, “*Why (teach) coding in the elementary grades and early years?*”

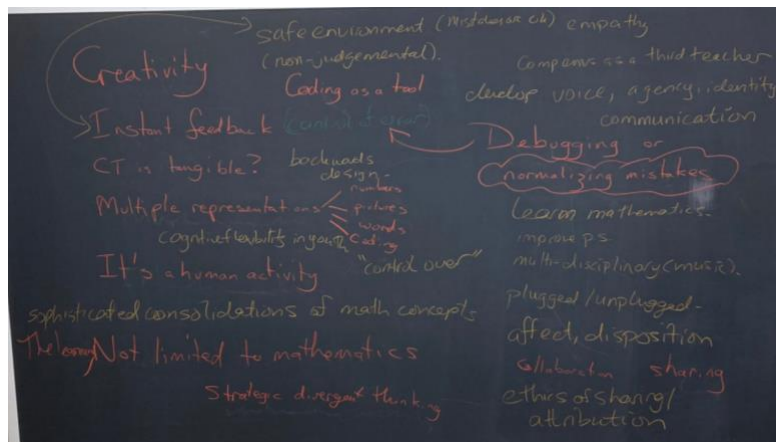
Responses to this prompt included:

*Coding is easier to learn in early grades; Coding opens doors for learners; Kids like robots and will like math; Coding encourages creativity and allows kids to create and problem solve while having fun; Coding normalizes mistakes and debugging as important disposition during problem solving; Coding provides multiple representations and strategies; Mathematics and CT are both human activities but also privileged human activities in our context and so*

*important to be able to use; Coding as providing opportunities for learners to develop and demonstrate voice, agency, identity and belonging; “Computational science is an offspring of mathematics, so its elements tend to involve sophisticated consolidations of ranges of mathematical concepts, contexts and representations; Coding as a ‘silent’ or third teacher drawing on Guy Brousseau’s ideas; Coding as part of the bridging connections between STEM and Mathematics Education allowing for interdisciplinary and contextually relevant learning (Figure 2).*



*Figure 2. Participants’ rendering of relationship between STEM and Math Ed with coding and CT part of the connections along with outdoor education, early childhood learning and global contexts.*

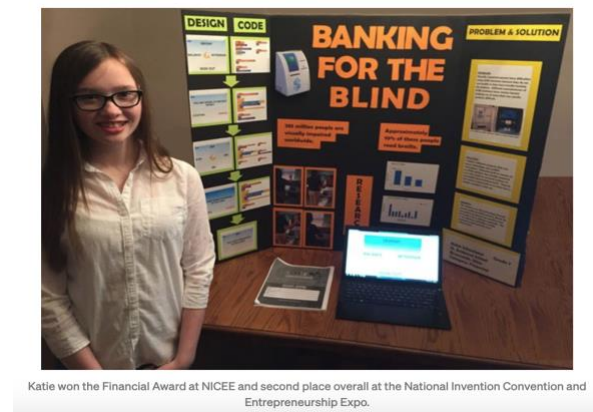


*Figure 3. Summary of some of the discussion about why teach coding.*

Gadanidis’ (2017) five affordances of coding for K-6 math were discussed. These include the following: Agency (opportunity to be in control of one’s learning), Access (by differentiation and ability to go explore more complex relationships and representations), Abstraction, Automation (allows automation of some processes), and Audience (shareable). Participants were also encouraged to consider whether they experienced any of these and where they had occurred during the working group.

In connecting to the Symposium theme on equity and computational Modelling and mathematics education, we shared two cases from the Scratch Team blog. The first (Nguyen, 2016) described Katie S., an 8th grade student who used Scratch to help visually impaired individuals to use ATMs using hand gestures. The second example (Singh, 2016) describes the case of Scratch user named Megaman100, a student who is neurodivergent, who developed confidence and technical skills with coding that led to the creation of his own video game characters. Both cases illustrate Gadanidis’ affordances and concretely foreground the way learning to code and working towards equity travel together. The Scratch organization also

presents several examples of the way the programming environment can be used by learners with disabilities (Adams, 2009). It is important to note that these examples do not include children in the earliest years and the examples are not strictly mathematical..



*Figure 4. Katie S. in front of her Scratch Project presentation: Banking for the blind.*

The second aspect of yearning asked what participants hoped to learn in the session (the YALL framework does not imply discrete categorization). Several themes emerged. One of the strongest was focused on teachers and wondered, “How can we build the capacity of pre-service and in-service teachers and guide them in understanding the relationship between mathematics and coding in a holistic way, i.e. to make connections between our knowledge in various fields and math/coding/CT” and to tease out the connections among coding, math and socio-emotional learning. This is a recurrent theme in the research literature over at least the last two decades. The second theme is also recurrent - “How are terms such as coding, computational thinking, coding, algorithmic thinking and mathematical thinking defined, and are they used with the same understanding by different individuals or groups?” Attempting to define computational thinking and mathematical thinking seems to be a necessary step in groups such as this to have productive conversations and is an aspect we picked up on the second day. The third emphasis was on tasks and framed using Dan Meyer’s (2010) now classic analogy repurposed as “What problems can we offer that are “headaches” for which mathematics AND code are the aspirin?”

## **Affirming**

Researchers (and teachers) have learned quite a bit about teaching and learning coding at various grade levels and the challenges and affordances of doing so in relation to mathematics. Research on coding in elementary settings and those focused on mathematics is more diverse than research on coding in the early years reflecting the different developmental and conceptual emphases. As a working group, we brainstormed what was affirmed about coding in the research and our own experiences, or essentially what the group already knew about coding. The main themes that emerged were that coding is characterized as the following:

- a vehicle for creativity
- A good source of productive failure or struggle
- marked by algorithmic sovereignty
- facilitates deep understanding about problem solving
- not computer science
- transforms objects of learning into tools for learning
- develops identity

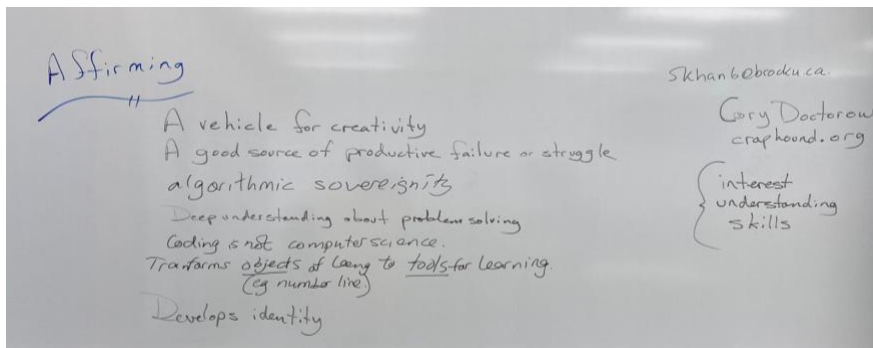


Figure 5. Working group responses about affirming what coding is based on research and our own experiences (or what we already knew about coding).

### Contexts Matter

One of the clear reminders that emerged over our discussions was that contexts matter a great deal and differences in contexts are directly related to equity (Gutierrez, 200?) for learners and teachers and researchers. In this section we present outlines of the different local (provincial) and individual contexts in which we work. In coming to appreciate why we might be talking past each other at times and in opening up space for epistemological plurality by recognizing our attentional blindspots due to hyperfocusing and assumed commonality, this work is necessary.

### Day 2

#### Learnings (Initial)

Day 2 began with working group members sharing some of the coding activities they engaged with in their professional settings. Coding activities about grids, goats, and games were shared with the group, ranging from early years to elementary level coding.

Table 1. Working Group Coding Activity Demonstrations: Grids, Goats, and Games

Name	Link	Description
Sandy	<a href="https://earlymath.ca/teacher-resources/roots-of-coding-series/">https://earlymath.ca/teacher-resources/roots-of-coding-series/</a>	Early years “Roots of Coding” grid activity called “X Marks the Spot” use to develop spatial language used during coding
Iain	<a href="https://scratch.mit.edu/projects/828890799/">https://scratch.mit.edu/projects/828890799/</a>	A beginning computational model with billy goats who eat the sweet, green grass - maybe too much. What is needed to balance this ecosystem? Could the troll help?
Iain	<a href="https://scratch.mit.edu/studios/31981209">https://scratch.mit.edu/studios/31981209</a>	Coin Flipping and More Oh, My! Exploring probability.
Alex	<a href="https://blockly.games/maze">https://blockly.games/maze</a>	Fun maze :) blockly has other activities too
Irina	<a href="https://www.birdbraintechnologies.com/">https://www.birdbraintechnologies.com/</a>	Resources for Finch robot
Irina	<a href="https://www.primotoys.com/">https://www.primotoys.com/</a>	Cubetto



*Figure 6. A Finch robot demonstration to our working group.*

Day 2 included opportunities to engage in rich discussions about the relationship between mathematics and coding. Topics included the following:

- What is mathematics?
- What is mathematical thinking?
- Does all coding include math?
- What is the relationship between computational thinking and maths?

Insights from our discussions included the following:

- It is important to make a distinction between proto-mathematics vs mathematics
- Coding itself does not bring rich math by itself, math concepts can be learned (engaged with) in the process of coding, but this needs to be intentional and made explicit by educators
- You can teaching meaningful math through coding and you can teach coding while students are learning math
- The word com-putare (computer) derives from Latin and means to take apart and put back together again
- This project is a civilizational transformation project; thus be wary of that

### **Using Commercial Puzzle Games and Puzzles to Develop Coding Related Processes**

Participants were introduced to several modern commercial puzzle games that can be used to develop coding related processes (computational thinking) across the early years through play, inquiry and guided discovery. Many of the games also depend on and develop skills related to spatial reasoning. All of the games involve elements of logical reasoning, sequencing, decomposition and provide multiple opportunities for debugging thinking, unlike screen-based coding environments.

Using games with a sense of computational awareness, i.e. with an awareness of computational thinking processes, allows teachers to work intentionally to ground the language of computational thinking with the concrete enactive experiences and productive disposition developed through game-play and puzzles. Opening up this dimension also opens the door to including many games from across cultures and time-periods - from Mancala to Hex.





Figure 7. Discovering how commercial games can be used to teach coding.

### Day 3

#### Learnings (Sub-groups)

On the third day of our working group, participants split into sub-groups based on their interests. Three distinct sub-groups were formed: 1) Computation tools for mathematics education, 2) The role of coding and computational thinking in mathematics education, 3) Building coding capacity in elementary teachers. Key learnings and wonderings are presented for each of the sub-groups.

#### **Sub-group 1: Computational Tools for Mathematics Education**

Sub-group 1 discussed and identified some helpful computation tools for mathematics education and created a flow chart to help education decide which computation tool to use.

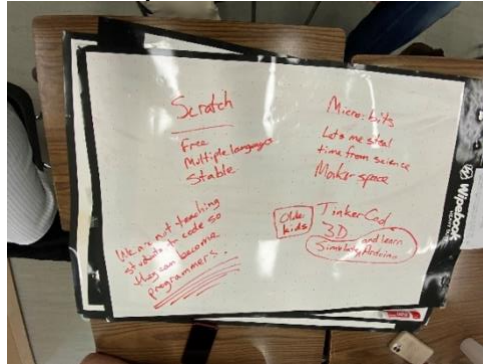


Figure 8. A list of computational tools that can be used for mathematics education.

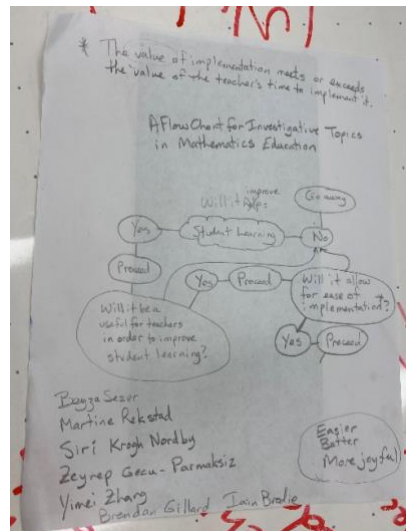


Figure 9. A flowchart to help educators decide which computational tool to use.

Sub-Group 1 concluded their time together by developing the action item and research question identified below.

Action: supporting teachers to use the tools effectively and value the benefits of the tools.

Research Question: How can we motivate students to use computational tools (i.e., especially robotics) to learn mathematics along with play (e.g., joy, engagement)?

### ***Sub-Group 2: The Role of Coding and Computational Thinking in Mathematics Education***

Subgroup 2 wrestled with the role of coding and computational thinking in mathematics education and the philosophical question about why we teach coding. Coding was discussed as a new literacy that can be used to promote culturally relevant pedagogy and promote greater equity among students.

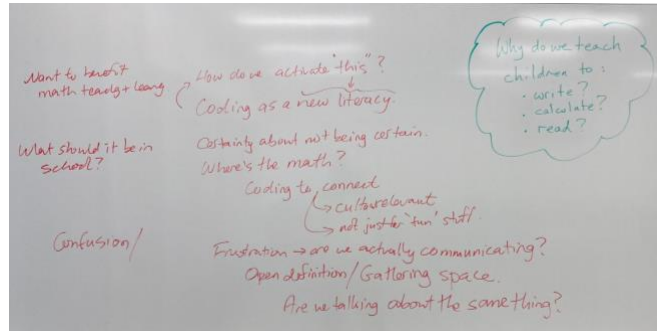


Figure 10. Guiding questions about the role of coding in mathematics education.

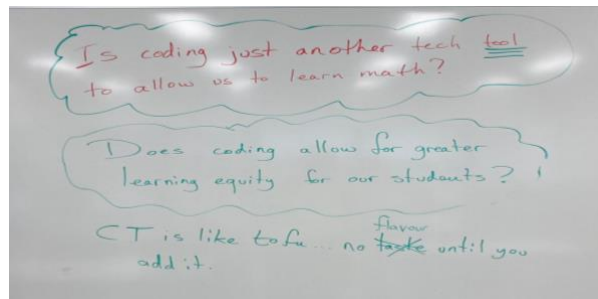


Figure 11. Further questions about coding and mathematics education.

Rather than coming to a consensus about the role of coding and computational thinking in elementary mathematics, Subgroup 2 developed the following questions for consideration:

- What do (interdisciplinary) rich mathematics look like with modern computational thinking in elementary school?
- What mathematics does modern computational thinking make irrelevant? (we commented about physicality of paper-pencil computation)
- What mathematics does modern computational thinking make important?
- What ways of mathematical knowing change as a result of modern computational thinking?
- How does modern computational thinking shift the time spent on relevant mathematical topics?

### ***Subgroup 3: Building Coding Capacity in Elementary Educators***

The building coding capacity in elementary educators brainstormed important ideas and strategies for strengthening teacher capacity. Based on these ideas, the sub-group identified the following insights:

- We need to develop as teacher educators ourselves and help pre-service educators become comfortable with learning with and from students (and us)
- We need to move beyond surface level professional development and coding tasks to equip students to be competent coding educators
- We can use the strands of mathematical proficiency to guide the development of a coding course for pre-service teachers

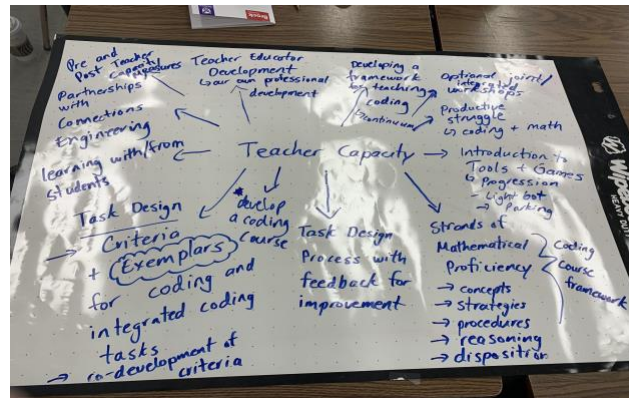


Figure 12. A list of Important ideas and strategies for building teacher capacity in coding.

Subgroup 3 decided on the following action steps:

- Develop a comprehensive and joyful framework for teaching coding
- Use tools and games that facilitate coding progression
- Work on high quality coding tasks and accompanying rubric with relevant criteria

**Note:** After the Coding Symposium, members of Sub-group 3 continued to meet monthly to develop a Social Sciences and Humanities Research Council Partnership Development Grant called, “Developing Coding Capacity with Elementary Educators”.

**Acknowledgements:** We wish to thank the Social Sciences and Humanities Research Council, the Fields Institute, and Callysto for making this Coding Symposium possible. Our working group greatly benefitted from the opportunity to meet and learn from one another.

### Final Thoughts/Reflections

There is a movement in education to include coding in the elementary classroom. We know that coding allows for the learning of skills that are useful in problem-solving, modelling, and could aid in some of the so-called 21st century skills like information and technology literacy, and critical thinking skills. A great deal of attention is being paid to just what type of thinking we are engaged in. Is it computational thinking we are engaged in (see Wing, 2006) computational modelling (see Gadanidis & Namukasa, 2011), computational fluency (see Resnick, 2018), computational participation, or computational literacy?

This is a really important discussion for the future of what and how we teach coding going forward, but it misses the mark for what is needed right now, at this very minute, in the classroom by the teachers who are delivering it to their students. More than anything classroom teachers need to develop deep content knowledge about coding and be introduced to effective instructional strategies for teaching it.





This is the arc that our working group took over the course of our three days together. We debated hotly about what type of thinking was going on when we code and slowly, but surely, we moved towards what teachers needed in order to implement coding in their classrooms. We eventually broke up into sub-groups with some taking on the important thinking about thinking and others working on what supports we need, and importantly don't need, to deliver to our colleagues in the classroom.

## References

- Adams, J. (2009). Using Scratch to engage students with disabilities.  
<https://scratched.gse.harvard.edu/stories/using-scratch-engage-students-disabilities.html>
- Bers, M. U. (2022). *Beyond coding: How children learn human values through programming*. MIT Press.
- Bers, M. U. (2020). *Coding as a playground: Coding and computational thinking in the early childhood classroom (2e)*. MIT Press.
- diSessa, A. A. (2018). Computational Literacy and “The Big Picture” Concerning Computers in Mathematics Education. *Mathematical Thinking and Learning*, 20(1), 3–31.  
<https://doi.org/10.1080/10986065.2018.1403544>
- earlymath.ca (n.d.). Roots of Coding. <https://earlymath.ca/teacher-resources/roots-of-coding-series/>
- Floyd, S. (2022). *The Past, Present, and Future Direction of Computer Science Curriculum in K-12 Education Electronic Thesis and Dissertation Repository*. 8463.  
[https://ir.lib.uwo.ca/etd/8463/](https://ir.lib.uwo.ca/etd/8463)
- Gadanidis, G. (2017). 5 straight A's for Coding + Math. *Math + Code Zine*, 2(3),  
<http://researchideas.ca/mc/straight-a/>
- Gadanidis, G., & Namukasa, I. K. (2011). New media and online mathematics learning for teachers. D. Martinovic, V., Freiman, and Z. Karadag (Eds.), *Visual mathematics and cyber learning* (pp. 163-186). Springer.
- Kafai, Y. B., & Burke, Q. (2017). Computational participation: Teaching kids to create and connect through code. In *Emerging research, practice, and policy on computational thinking* (pp. 393-405). Springer.
- Meyer, D. (2010). Math class needs a makeover. TED Talk  
[https://www.ted.com/talks/dan\\_meyer\\_math\\_class\\_needs\\_a\\_makeover?language=en](https://www.ted.com/talks/dan_meyer_math_class_needs_a_makeover?language=en)
- Microbit.org (n.d.) <https://microbit.org/teach/for-teachers/>
- Nguyen, M. (2016). Meet the Scratcher: Katie S. Medium.com.  
<https://medium.com/scratchteam-blog/meet-the-scratcher-katie-scheutzow-18464f757f64>
- Pérez, A. (2018). A framework for computational thinking dispositions in mathematics education. *Journal for Research in Mathematics Education*, 49(4), 424-461.  
doi:[10.5951/jresmetheduc.49.4.0424](https://doi.org/10.5951/jresmetheduc.49.4.0424)
- Resnick, M. (Sep 2018). Computational Fluency. Medium (adapted excerpt from Lifelong Kindergarten).
- Sinek, S. (2009). How great leaders inspire action. TEDx  
[https://www.ted.com/talks/simon\\_sinek\\_how\\_great\\_leaders\\_inspire\\_action?language=en](https://www.ted.com/talks/simon_sinek_how_great_leaders_inspire_action?language=en)
- Singh, S. (2016). How Scratch changed the life of our son with autism. Medium. Com  
<https://medium.com/scratchteam-blog/how-scratch-changed-the-life-of-our-son-with-autism-a0b9ef0f6388>
- Wing, J.M. (2006) Computational thinking. *Communications of the ACM*, 49, 33-35.



## **Coding, Computational Modelling & Equity in Mathematics Education Working Group B Secondary/University Report**

### *Working Group leaders*

France Caron<sup>1</sup>, Steven Floyd<sup>2</sup>, & Miroslav Lovric<sup>3</sup>

<sup>1</sup>Université de Montréal; <sup>2</sup>London District Catholic School Board; <sup>3</sup>McMaster University

### *Working Group Members*

Chantal Buteau  
Sarah Castle  
Amenda Chow  
Andy diSessa  
Francis Duah  
Wendy Forbes  
Krista Francis  
George Gadanidis  
Bogumila Gierus

Oh Nam Kwon  
Kehinde Ladipo  
Brian Lawler  
Gabriel Lecompte  
Haobin Liu  
Elise Lockwood  
Neil Marshall  
Ariane Masuda  
Simon Modeste

Eric Muller  
Mohammad Reza  
Peyghami  
Elena Prieto-Rodriguez  
Ana Isabel Sacristán  
Jessica Sardella  
Pam Sargent  
Marie-Frédéric St-Cyr

### **Introduction**

Mathematics is used in designing and coding algorithms (Modeste, 2016), and coding is often used to expand the class of mathematical problems that can be solved, or the set of concepts that can be explored (Buteau et al, 2020; Lovric, 2018). Mathematics and coding can also be used jointly with principles from other disciplines to model real-world situations (Giabbanelli & Mago, 2016; Caron, 2019).

In this working group, we examined coding and computational modelling in secondary and university mathematics by working with, and developing, hands-on activities for students and educators. We explored ready-made programs related to secondary and university mathematical concepts, along with tasks that have been, or could be, built from these programs, and then considered the characteristics of effective program-task combinations.

Important concepts, practices and ideas associated with computational thinking, computational modelling, mathematics, or computer science were identified in the effective tasks and activities. As we developed our ideas in discussing activities and teaching, we aimed at connecting them to, and contrasting them with, existing research and associated frameworks (Weintrop et al., 2016; Grover & Pea, 2017; diSessa, 2018; Modeste, 2018; Bråting & Kilhamn, 2021; Dohn, 2020). The intent was to identify key elements that could be made part of a foundation to create new activities, as well as envision improved integration of mathematics, coding, and computational modelling instruction.

We invited participants to bring, if they wished, their own samples of coding or programming tasks.

### **Participants and Prepared Activities**

Prior to the working group meeting, a form was sent out to the working group registrants seeking background information. The data received through this form indicated that participants were a diverse group that included high school and adult education teachers, college and university instructors, researchers, and graduate students. When asked to describe their experience with



coding and computational modelling, their responses ranged from limited and beginner to strong and "I have taught coding and computational mathematics". In terms of experience with specific platforms and languages, 52.9% of respondents had some or a lot of experience with the Scratch programming platform, while 63.2% had some or a lot of experience with the Python programming language. The responses from the form also indicated a variety of experiences with different programming languages and platforms including R, Matlab, Visual Basic, Java, SQL, Lego Robotics, C++, Mathematica, SAS, SPSS, Octave, and others.

In preparation for the Brock meeting, the facilitators created a set of over 25 mathematics and computer programming based activities, to serve as examples and starting points for discussions. Some of the activities addressed contemporary, real-world problems such as large language models and encryption in cyber security applications.

As well, the facilitators organized an excel file that categorized and summarized the various activities, so that the participants could access those that were most relevant to their work, and those that they were most comfortable with, in terms of mathematics and computer programming. In order for participants to decide which activities were most relevant to them, the excel file included information related to the programming language used, the mathematical topic and the approximate school level of mathematics where it could be used, as well as the levels of programming involved from L0 (simply observing the results of the running program) to L5 (designing and writing all components of the program) based on Broley et al.'s (2017) categories.

When selecting a particular activity, participants had access to more detailed information which included, as shown in the table below, the mathematical and computational concepts as well as some habits of mind associated with either discipline that could be mobilized in the activity. Many of these habits of mind have been identified in research papers (Weintrop et al., 2016; Grover & Pea, 2017; diSessa, 2018; Modeste, 2018; Bråting & Kilhamn, 2021; Dohn, 2020), which were made available to the participants; using them helped convey the idea that a single activity could benefit both mathematical thinking and computational thinking, while offering potential for connecting ideas and concepts from both disciplines.

**Example:** Learning scope of the Linear Combination Activity

	Mathematical	Computational
Concepts	Vectors, points, translation Linear combination Linear independence Uniform distribution (probability)	Loop Random generator Variable types and identifiers Trace
Thinking / Habits of mind	Generalizing Explaining	Debugging Generating test cases Validating

The activities were developed using a variety of "computational tools", including Excel, GeoGebra, Insight Maker, Python, Scratch, and Excel. All activities that were developed included guiding questions and extension activities (or the next steps) so that participants could complete the initial activity, and also extend their thinking to see how the activity could be modified or enhanced.

The group met for five working sessions, from 90-120 minutes each, over the span of three days, and culminated in a final, sixth, session where the work was presented and summarized for

all those in attendance at the symposium. The diagram below outlines the dynamics of our working group activities over the three days of the Symposium:



### Objects (and Questions) to Think (About What) With

In *Mindstorms*, Papert (1993) discusses how physical gears were important in his early development as a mathematician, as these served as a way to help mathematics enter into his life at a young age. He explains how the gears served as models, carrying abstract mathematical ideas, and how they allowed for Piaget’s (1952) notion of assimilation to take place, whereby new information fit into existing cognitive schemas within his mind. He describes these gears as “objects-to-think with”.

When developing his physical, programmable turtle for students, which was controlled using LOGO programming code, Papert was attempting to create another “object-to-think with”, that could serve the same role for students as the physical gears had done served for him: “My interest is in the process of invention of "objects-to-think-with," objects in which there is an intersection of cultural presence, embedded knowledge, and the possibility for personal identification” (Papert, 1993, p. 11).

In our working group, participants were provided with a number of small mathematics and programming activities that served as “objects-to-think with” during the three days we were together. The activities included a range of complexity in terms of mathematics and programming concepts, and they also were developed to present a variety of digital tools that can be used (e.g., python or scratch programming language, Insight Maker). The activities were presented to participants at the start of the first session, and participants had time to work with the activities, discuss them, and extend them. The following day, participants were invited to share their own activities, or to share how they extended provided activities.

Much like the gears of Papert’s childhood, or his physical, programmable turtle, the activities in this working group served as “objects-to-think with” as they included an intersection of cultural presence, embedded knowledge, and the possibility for personal identification. Participants were able to manipulate the activities by altering the code, providing an opportunity for assimilation to take place, whereby this new information (mathematics, programming, or teaching related) could potentially fit into existing cognitive schemas within the participant’s minds.

As an example, participants were provided with a short snippet of Python programming code related to the story of Anno’s Seeds (Anno, 1995). This story, and associated computer program, is summarized and discussed in Gadanidis, Hughes, Namukasa, and Scucuglia (2019). The story involves Anno being given two magic seeds that, when eaten, can sustain him for a year. He decides to eat one of the seeds, and plant the other. A year later, the planted seed has produced two new seeds. He again eats one of the seeds, and plants the other, and repeats this year after year.

After writing and running the code that models this story, participants were then asked to alter the code, so that instead of eating one of the seeds at the start of the year, Anno eats one of the seeds after they have been planted and after they have yielded new seeds.



As participants worked in pairs or in small groups in altering the code, conversations took place that focussed on the syntax, purpose and components of each line of code (programming aspects), the potential growth in the number of seeds (mathematical aspects), and in how a program like this could be used to support student learning (pedagogy and andragogy aspects). Participants were also asked whether there were other mathematics stories, similar to Anno's seeds, that could be modeled with code.

As participants worked to alter and extend the Anno's seeds example, the conversations revolved around the small snippet of code itself, thereby allowing it to serve as an "object-to-think with". After all participants worked with the Anno's seeds example, they were then given the opportunity to explore over 15 other activities, each serving as a basis for conversations and learning, and ultimately each serving as an "object-to-think-with". The thinking did not limit itself to the mathematics involved. For instance, a participant created a sequence that could generate negative numbers of seeds, in which case she wanted the sequence generation to stop. Moving from mathematical thinking to computational thinking, and knowing that a go to statement is not the best for readability and adaptability, she created a while loop that checked for the negative value condition.

When considering how to best plan future coding and mathematics working groups or experiences for educators, we would recommend providing participants with coding and mathematics activities that serve as "objects-to-think with". We found that this facilitated conversations and discussions throughout the time together, as participants referenced specific code, mathematics, or instructional components from the activities. It also provided the group with a wide range of hands-on, learning activities upon which to build.

### Case Studies

We discuss two of the activities that the working group participants engaged with. In the *One-dimensional random walk* activity, we use simulation and computing to investigate a mathematical idea involving randomness, and to arrive at an important conclusion. In *Cryptography*, our mathematical knowledge of modular arithmetics guides us in creating and modifying a Python code that is used to encrypt and decrypt messages, thereby showing an application of mathematics in computer science.

**Investigating one-dimensional random walk** activity was designed to encourage and invoke questions about mathematics, which can then be verified using code (as is, or modified).

Brief description: A particle starts at  $x=0$  on a number line. In each step, with equal chance, it either moves left or right for one unit (thus, after the first step, the particle is either at  $x=-1$  or at  $x=1$ ; if it is at  $x=1$ , in the next step it will move to  $x=0$  or to  $x=2$  with equal chance). Describe what happens after 2, 3, 4, ...,  $n$  steps of this random walk.

The motivation behind this question is important. Albert Einstein used the random walk model (in two and three dimensions; also known as Brownian motion) to describe a diffusion process, and to convince "reluctant physicists to accept the existence of atoms" (*Annus mirabilis papers*, Wikipedia). In our model, the left-right movements of a particle are due to collisions with other particles.

A student or a teacher could suggest another situation where this model could be applied. Indeed, one of the members of our WG said: "The random walk example can be analyzed by students, and then they can develop a simpler program to simulate rolling two die. They can then graph the distribution of the rolls. All of the code needed is included in the random walks example. This is perhaps reverse engineering." (This remark was given in the context of an interesting





suggestion: “We often provide students with a simple program, and then have them extend and build upon it. What about working the other way? Providing a relatively complicated program, and have them use the code within the program to build something relatively simpler.”)

The activity sheet encouraged participants to work with a pencil and paper (or a notebook) first, before coding. This is an important step, as thinking about the problem will suggest important mathematical questions, such as: What is the farthest location from the origin a particle can be after 2, 3, 4, ..., n steps? What is the range of the ending locations after n steps? Is there a pattern? Are all ending locations equally likely to occur? Moreover, deep(er) understanding of the mathematical problem at hand suggests a logical structure for the code to solve it (which is an essential step in coding!).

Once a student understands the code and starts using it, many doors of investigation open. The activity sheet suggests some: Pick a certain number of steps (say, 20) and run the simulation 10, 1000, and 10,000 times to obtain a distribution of ending locations. Is it what you expected it to be? Investigate what happens when you modify the probabilities (so that the particle is ‘biased,’ i.e., there is a higher chance of the particle going in one direction than in the opposite direction). Can you predict what the distribution will look like in this case? Modify the code appropriately and run the simulations to see if your prediction was accurate.

Here is the code that was given to our WG participants:

```
1 # random walk in 1D
2 # import relevant modules
3
4 import matplotlib.pyplot as plt
5 import random as ra
6 %matplotlib inline
7
8 # simulating one-dimensional random walk
9
10 numsteps = 6
11
12 x = 0 # initial location
13
14 for i in range(0,numsteps):
15     a = ra.randrange(1,3)
16     if a==1:
17         x=x+1
18     if a==2:
19         x=x-1
20     #x = x + (-1)**a    CHALLENGE - what does this line of code do?
21     print("time:", i+1,"location:",x)
22
23 print()
24 print("The final location of particle after", numsteps, "steps is", x)
```

The participants who engaged with this code were able to go through it and gain basic understanding; for instance, anything in a line that starts with a hashtag is a comment (documenting code is important, as it facilitates its understanding!) and is ignored by Python; certain modules (accepted here as black boxes) have to be imported, as they contain routines and commands that will be used; the word ‘for’ at the start of a line indicates the start of a loop, used to repeat certain actions; etc.). Certain commands needed to be explained; for instance, it is not at all obvious (nor logical) that `ra.randrange(1,3)` generates a random number from the set  $\{1,2\}$ , where one number is equally likely to be selected as the other. Once this is resolved, the sequence of two ‘if’ commands is clear.

An interesting math question is the following: can you make the code shorter, by removing the ‘if’ statements? That was the point of the challenge in line 20, which, unfortunately, our participants did not engage with.

Here is a sample run of the code:

```
time: 1 location: 1
time: 2 location: 0
time: 3 location: 1
time: 4 location: 0
time: 5 location: 1
time: 6 location: 2
```

The final location of particle after 6 steps is 2

Of course, the most natural thing to do is to keep running the code and observing what happens. The next extension, suggested by the activity sheet, is to make this process automatic. At this point, a student has already seen one basic element they need in terms of coding, i.e., the loop. In our view, this is an important step in learning to code: you have seen something work, how can you use it again? One new element is needed: every time a random walk is run, we would like to record the final location. For this purpose, a list is used. How does a novice programmer learn that there are lists, and how to work with them? Here, we argue that some course instruction can help, or, a hint such as “look up lists in the Python manual that we have been using.”

Here is one way to automatise repeated random walks:

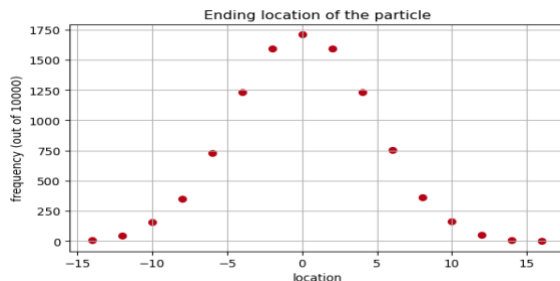
```

1 # simulating multiple random walks, all with the same number of steps
2
3 numsteps = 25
4 numwalks = 20
5 xend = [] # to save final locations of the particle
6
7 for walk in range(0, numwalks):
8     x = 0
9     for i in range(0, numsteps):
10        a = ra.randrange(1,3)
11        x = x + (-1)**a
12        xend.append(x)
13
14 print('Final locations of particle:', xend)
15 print('Farthest location on the right is', max(xend))
16 print('Farthest location on the left is', min(xend))

```

Final locations of particle: [11, 5, 5, -9, -1, -3, 1, -1, -11, 1, -3, 3, -5, -7, 7, -1, 3, 1, 3, -5]  
 Farthest location on the right is 11  
 Farthest location on the left is -11

Final locations are recorded in a list, and then, with the help of statistics and graphing (initially accepted as black boxes, but later, as one learns more about coding, some of their features should be revealed) Python generates the following diagram of ending locations of 10,000 random walks with 20 steps each:



As a further extension (“wide walls”), we find a suggestion from a participant: “This code can be used to teach students about all systems in which a transmission from one state to another one is completely random. To add a bit fun to this topic, I would design a system for which the random walk can be examined in the class. For example, assume that seats are the states of a system and I ask everyone to run the code for a number of steps (let say 10). Assuming that the initial position is the student’s current seat, we set up some rules to change the seat based on the code’s output. For example, the movement in the simple case would be stepping forward (+) or backward (-). Then, we can ask them to identify their location at the end of the day. We can further change the design to add more randomness to this walk by for example alternating the movement between up/down and right/left.” The last sentence suggests modelling a two-dimensional random walk!

The activity **Cryptography** was designed to engage students with a fun objective (to create and then to break a code), together with working on a mathematical topic of *modular arithmetic*. Basic *encoding* and *decoding* techniques were considered: translation (also known as the Caesar code) and encoding with a key, based on the arithmetic modulo 26. A unique number needs to be

assigned to each letter of the alphabet (to simplify, we use uppercase letters). The most convenient way is this: A=0, B=1, C=2, ..., Z=25. In Caesar code, each letter is moved the fixed number of locations in the alphabet to the right. Coding with a key (also known as Vigenère code) involves “adding” a letter and the corresponding value of a key.

As this activity is - in terms of coding - somewhat demanding, participants were given the entire code to start with (of course, they needed to code extensions and their own ideas). Several participants commented on the following piece of code, which establishes the mapping A=0, B=1, C=2, ..., Z=25:

```
1 ##### convert a letter to a number and back (A=0, B=1, ... Z=25)
2
3 def letter_to_code(st):
4     return ord(st)-65
5
6 def code_to_letter(n):
7     return(chr(n+65))
```

For instance: “I enjoyed exploring the Cryptography task. I would say that I tried to dig into the code a bit to understand it and explore it. I was at first unsure about the need to subtract 65 in the conversion between letters and numbers, but I googled (with the help of a partner) the ord() command to learn about Unicode and the fact that A is 65 in Unicode.”

Another participant echoed this remark: ““Playing with the code was very fun and interesting. I understand why you give “ready-made” programs to work with, as many “tricky” things are used in some parts, that could be technical and difficult (and not very interesting) for students (examples: The chr(n+65) trick using ASCII codes; clearly it is quicker than creating a correspondence table or a converting function with 26 cases ...)”

About stimulating thinking, and switching between coding and modular arithmetic, one participant wrote: “The task was a joy. It challenged me mathematically and with coding. I appreciated having to think in terms of modular arithmetic. [...] really helped me frame my thinking. I was fortunate to have two teachers [...] to help me with working back and forth between the mathematical concepts and translating them into coding a solution to the problem.”

One participant commented on the virtues of working with a colleague (another important approach to learning in general, not just coding): “ ... and I discussed with a partner why the mod 26 was necessary and what it was doing. It was nice to discuss that with a partner and to think about where the letters are in unicode and why the mod 26 was necessary (we also discussed why the mod would work regardless of subtracting or adding some shift).”

Some participants added their own comments, etc, to make the code textbook-like. Some analyzed the underlying mathematics, drew a circle to represent the periodic behaviour when calculating modulo 26.

### **Considering and Combining Multiple Computational Approaches**

As reflection of where we are (or where we appear to be heading) with respect to coding integration in mathematics education, there was a keen interest from most participants in engaging in math learning activities with Python. About half of the participants had coding experience with it; those who did not had an opportunity to learn some of its elements through our activities. We thus had a substantial amount of activities in Python.

As we wanted to enlarge the discussion to other programming environments, derive similarities, point to specific uses and strengths, and even revisit what we mean by coding, we included activities that made use of Scratch, Excel, GeoGebra and Insight Maker. In the same way that multiple representations have been valued to learn mathematics, we aimed at showing the



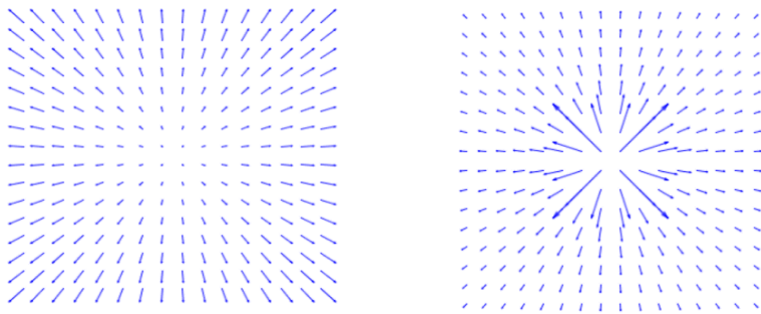
complementary contribution of multiple computational approaches for looking into a common situation or problem.

One of the key computational concepts is iteration (or loop), which can be linked quite naturally to the mathematical concept of a sequence, which itself can be connected to the concept of a function, as a discretized version of it. This was brought to light with the first introductory Python activity (Anno’s seeds), where a simple change in the order of operations led to creating two very different sequences (and functions). The generation of these sequences leads to data structure consideration and thus makes us move to a type of thinking that is more computational than mathematical.

The first version of the code had each sequence generated within a single scalar variable where a value, once generated and printed, would be replaced with the next value of the sequence (with the troubling-at-first use of the equals sign in assignment instructions such as  $x = x + \dots$ ). Graphing a sequence however requires keeping in memory, and in the right order, the different values of the terms of the sequence. As could be seen in the second version of the code, this can be done by generating two lists: one for the ranks (corresponding to the years here), and one for the values of the corresponding terms in the sequence. Such lists can be dynamically extended by appending new elements. A simple call to the plot function of the pyplot module, with those two lists as parameters, will generate the graph. Format options can be specified.

List generation by appending successive items can also be done with Scratch. However, graphing in Scratch requires to manage at a lower level the association between the rank and the value of its associated term (or between the x and y values of a function) and to direct the software to “stamp” a “dot” for each of these pairings in the 2D cartesian space provided (or to move with the «pen down» between two “consecutive” points of a function). This may in turn require scale conversion to adjust the target graph to the available space. In an interesting way, programming in Scratch may help unpack the black box that graphing can be to students and have them appreciate at a deeper level both the connection between a table of values and a graph and the necessary discretization and iterative process for graphing a function. As testified by one participant, young students may spontaneously elect to “look inside”: given a graphing tool, they may prefer to figure out how to store the data of a graph and use the idea for their own work.

As was suggested with one of our activities, the approach can be extended to graphing vector fields, with embedded loops for running through a discretized 2D space. Given the Scratch code that produced the first figure below (on the left), a team of participants challenged themselves to produce a vector field that could represent a repulsive force whose magnitude decreases as distance increases, as shown in the second figure on the right.



Producing the code for generating similar arrows (where the length and the head size are proportional to the norm of the vector) that will point in the right direction can be an interesting exercise of vector geometry and linear algebra, which can be done without requiring angle

computation and use of trigonometry. Prototyping it in GeoGebra (or having to explain a ready-made GeoGebra construction protocol for that purpose) can prove a useful preparatory step. It may also suggest that such a sequence of construction steps, applicable to an arrow of any size and direction, could be considered a form of code.

Building on the ideas of iteration and sequences, a set of four short programs had some participants move into the realm of numerical methods, through exploration of converging sequences defined by recurrence relations.

<pre> M # Familiarly Converging Program  a = float(input('Enter "random" number: ')) print(a) x=a/2 print(x)  for i in range(10):     x=(x+(a/x))/2     print(i+1, x) </pre>	<pre> M # Mysteriously Converging Program Version 2  a = float(input('Enter "random" number: ')) print(a) x=a/2 print(x)  for i in range(20):     x=(2*x+a/x**2)/3     print(i+1, x) </pre>
<pre> M # Mysteriously Converging Program  a = float(input('Enter "random" number: ')) print(a) x=a/2 print(x)  for i in range(20):     x=(x+a/x**2)/2     print(i+1, x) </pre>	<pre> M # Stubbornly Converging Mystery Program  import numpy as np  a = float(input('Enter "random" number: ')) print(a)  for i in range(50):     a=np.cos(a)     print(i+1, a) </pre>

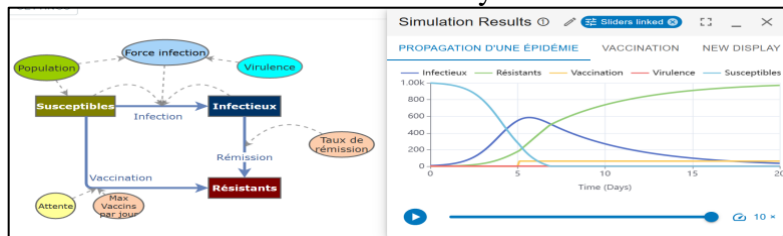
By running each program a few times with different input values, participants could try to infer the connection (if any) between that input value and the number towards which the output sequence appears to be converging. As these sequences are all defined by recurrence relations of the form  $x_{n+1} = f(x_n)$ , despite being written as  $x = f(x)$ , the key strategy for finding and explaining the limit of the sequence is actually to express (and possibly simplify) the solution of equation  $x = f(x)$ . Unveiling what these programs do, and mainly how they do it, allows students to envision algorithms that can serve as viable alternatives to the black boxes associated with magical function keys of their calculator (or preprogrammed functions of a programming language) such as square root. Running the two versions of the “mysteriously converging program” shows two algorithms converging at different speeds to the same limit, and can open to the concept of rate of convergence. The “stubbornly converging mystery program” can serve as an entry point into cobweb diagrams.

In playing with these programs, one participant raised a highly relevant question: “What is the added value of Python here when compared to Excel?” There may not be a single answer to that question, as it may very well depend on what the students already know, the learning objective and the connections that can be made in moving from one environment to the other. Despite the smaller distance with mathematics that the Python syntax and reference to variables may appear to have when generating sequences, the access to the explicit definition of each cell in Excel and the ease of replicating relative references may provide a more concrete handle at first, from where to build generalization and understand the idea of iteration.

Maintaining the possibility of comparing affordances for learning, participants were invited to explore dynamical complex system simulations, epidemics in particular, with different environments: Insight Maker, NetLogo, Scratch, Excel and Python.

Like similar software, such as Stella and Vensim, Insight Maker supports a system dynamics approach to modelling complex systems, with an icon-base interface for defining stocks and flows. Stocks are used to represent variables (e.g. number of infected persons) that accumulate over time (as integrals) and are given an initial value, while a flow changes a stock at every time step, either by adding to the stock (inflow) or subtracting from the stock (outflow). The flow (e.g. the rate of

infection) corresponds to a rate of change (or derivative) that can evolve over time, depend on other variables, and be defined with functions and even, as is the case with Insight Maker, with programming instructions. For example, in the model below, the vaccination flow (that has people move directly from the susceptible to the resistant group) has been modeled crudely with a conditional statement that assigns it a constant daily value if the number of days since the beginning of the epidemics is greater than the waiting period (before a vaccine becomes available and effective) and leaves it to zero if not. Students can be invited to think of more sophisticated models that could better reflect reality.



As has been observed in studies with students (Doerr, 1998) and with our participants, modelling with the stock-flow approach can be unsettling at first. In order to facilitate the understanding of the hydraulic metaphor behind this approach, a team of participants envisioned an activity where students would describe, compute and compare the amount of water used in a shower and a bath. Spending time to model and implement a filling bathtub (and envision a way to avoid overflow with a conditional statement) has also been found to be more than useful in making secondary school students experience a first success with Insight Maker and be more open to debugging when a modelling mistake or syntax error prevents their model from functioning adequately (Gonczi et al. 2022).

Once students have made sense of the stock-flow dynamics, it can help them expand radically the class of situations and scenarios that they can model and simulate. Although it may not be perceived that way by students that learn to model with it in their secondary science class, the approach provides a visual representation of a system of differential equations, which can be solved (or numerically integrated) using the simple Euler's method. This is the black box upon which the simulation is based and that must be opened at least once for students to maintain control over such software. This can be done by using Excel or Python to implement the simple numerical scheme to generate the sequences associated with the evolution over time of each of the variables. Examples of that were provided to the participants. One of them commented: "I can see uses for this in my first year university calculus course with connections to derivatives, especially related rates and introduction to differential equations. Of course it is also very useful for a differential equations course."

Dynamical complex systems can also be described with agent-based models (Caron, 2019). In such models, agents are programmed to move in a space, interact together, and change state depending on their interactions and the time elapsed. Elements of randomness are typically included in such descriptions. NetLogo is a multi-agent programmable modelling environment, designed to be "low threshold and no ceiling". It has been used with students, and is increasingly being used by scientific researchers with a limited programming background. Scratch and Insight Maker also support agent-based modelling. Short programs in each of these environments were offered to participants to experiment with agent-based computational models (of epidemics and flocks of birds), to see the effects of different parameters and infer the meaning of the coding instructions in those environments. Those who tried them felt that it was rather difficult to

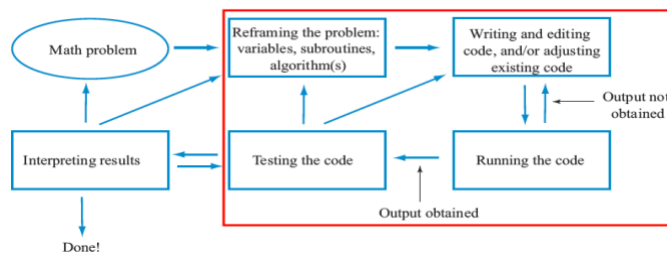
understand the modelling code in a programming language that they did not know, even with the possibility of comparing with a similar program in another language.

The fact that a model is often built on principles that come from a discipline other than mathematics and computer science certainly adds to the difficulty. We recognize that learning math, modelling, and programming at the same time involves a steep learning curve and that teaching should try to provide appropriate scaffolding. Yet, one should not neglect the inherent difficulty of adapting to a new programming language or paradigm (e.g. from procedural to object-oriented programming). This was also expressed in our introductory activity by a participant who was new to Python and the coding environment that we used.

### Python Notebooks as a Coding Environment

There is a wide variety of platforms available for coding, varying from the ones for beginners to those for professional coders. The platforms used for educational purposes are often easy to access, and straightforward to use. For Python, a convenient way is to use notebooks. One such setup is Jupyter notebooks (<https://jupyter.org/>), made available to researchers and students at Canadian institutions through the *syzygy.ca* service, created in collaboration between PIMS, Compute Canada and Cybera in 2017. A notebook is a collection of cells, where each cell can be executed on its own.

Perhaps the most appealing feature of the notebooks is that we can create, test, and run small parts of a code in one cell, and when we are sure that the code is working, we can move it to another cell where (often) a larger code is built. In terms of a coding model (Lovric, 2018; p. 689), working with individual cells within a notebook facilitates the important steps in coding, in particular the ones within a red rectangle:



One of our WG participants echoed this comment: “I also played around with just doing one of the programs at a time (in particular, being able to enter the encoded text and decoding it, and that presented some interesting opportunities for debugging).”

Another participant asked: “One of the questions that I had was about the jupyter notebooks we were given. Were these originally separate scripts?” The answer is - yes, originally these were separate scripts. Whenever faced with writing a complex code, a good way to break it down into simpler, smaller parts, and write a code for each part (modular approach). The most important reason why we do this is testing: if we write the entire code and Python returns an error, or the output is not correct, it is often not easy to find the source of the problem.

Notes: The values defined in one cell are “remembered” by Python until they are altered (by another assignment command) somewhere else in the notebook. Once a notebook is closed, all values are forgotten. A fully functioning piece of code (a module, created in a cell) can be (and often is) used, with confidence, in coding other problems. Python allows for comments (either inside a cell with code, or in a separate, text-only cell), which is very important, as we need to document the code we’re working with.



## Considerations Related to How Much Code is Provided

In teaching coding we need to make important decisions, including answering the following questions: Should we provide code (in exercises, assessments), and if so, how much of it? What pieces of code should we include? What do we expect our students will do with that code? In the absence of research results about these questions, we rely on our personal/professional, classroom, and other experiences.

Perhaps the best answer is - how much (if any) code is revealed (given to students) depends on the situation (topic(s) covered, teaching approach, pedagogical considerations, and so on). If we wish to provide a “low floor” approach, and/or work with novice coders, giving some code is a good idea. For instance, it could be a functioning code that needs to be refined and/or modified, or several starting lines of a code which define the variables or suggest a logical structure.

Seeing sample code helps students with the non-intuitive, confusing at-first elements of coding (and an initial barrier to learning!), such as learning what the names of commands are, what the commands do, and the syntax. For instance, to ask Python to show the graph of a function, we need to use `plot` rather than `draw`; the command `range(0,4)` generates the list of integers 0,1,2,3 which includes zero but does not include 4; a list is indexed starting from 0 (i.e., we refer to the first element in a list `L` by `L[0]` and not `L[1]`); a code will not work if the `for` statement opening a loop does not end with a colon).

Providing code (as well as appropriate comments documenting the code) helps our students learn from professionals, and do as professionals often do, which is a productive way to code. When we teach proofs in mathematics (perhaps the most difficult concept for students to learn), we start by showing to our students how proofs work, and explain each proof step-by-step. We direct their attention to the logical structure of a proof, to the narratives that are used to help the reader follow the argument, and so on. This is a challenge to absorb at once, it takes time to master. As students advance through their studies, they slowly start to create their own proofs, first with assistance, and later on their own. Learning to code is, in many ways, like learning proofs (or mathematics in general). This process is mimicked in the so-called PRIMM (predict-run-investigate-modify- make) approach to learning to code, where a learner moves from a code which is not theirs (predict, run, and investigate), to the one which is partially theirs (modify), to the one that they create (make).

On the other hand, our anecdotal evidence suggests that providing code can be counterproductive, as it might obscure the coding process and reasoning. For instance, a student might miss a reason why a certain variable was coded as a list, or why a certain loop was used to implement an algorithm. Similarly, a math proof that starts by announcing a method used (“We will prove this statement by contradiction”), does not give an opportunity for a student to figure out why that particular method is used.

On this theme, our participants offered their own thoughts and asked questions:

- “We often provide students with a simple program, and then have them extend and build upon it. What about working the other way? Providing a relatively complicated program, and have them use the code within the program to build something relatively simpler.”
- ““Also with the examples a lot of these have us modifying the coders initial thoughts. But why couldn’t students first start with the problem such as cryptography and the caesar key and go through the process of building up this formula?”
- “But clearly, we see that this very simple cryptographic principles, Caesar and Vigénère codes, are not so easy to implement and you really get confronted with programming





issues in order to implement them, and from a blank page, students would be in difficulty. But, at the end, creating a cryptography program from scratch is what we would like students to be able to do, isn't it?"

In summary, there are two types of decisions we need to make when teaching coding:

- Code: Do students begin with a ready made code? Do we encourage them to copy and paste pieces of code, or write their own? Or, do we give them a starting part of a code, where we define the variables to be used and suggest the approach? Or, do we start with an empty cell/ empty notebook?
- Complexity: How do we stimulate students to break up a complex problem into smaller, easier-to-code pieces (modular approach)? Does it make sense to give to students a relatively complex program, and then use parts of the code to build something simpler?
- Of course, these decisions are of dynamic nature - they need to be made or modified on a daily basis, depending on the students' progress, and the content which is taught.

### Conclusion

Integrating coding in the learning of mathematics opens possibilities for exploring, understanding and modelling, but it does require addressing and resolving tensions that may arise when making decisions on the programming language, the coding environment, and the code made available. As time is a limited resource in any class, the time spent on learning a new language or even in writing original code may appear as time taken away from learning mathematics. This (and the impression that it was probably hard to convince math instructors that coding has a power to enhance the understanding of mathematics) is why programming in the math classes had been progressively replaced in the 90s by user-friendly applications. Although such applications have provided new means for teaching, learning and doing mathematics, they often came as black boxes with a limited scope of use.

Learning to program can help open these black boxes and extend the scope of application of the mathematics we learn. To fully reap such potential, we may have to consider not only *coding for mathematics* as another way of learning mathematics, but also *coding with mathematics* for modelling and understanding real-world situations. This might bring together more mathematicians, computer scientists and others scientists in helping address the current challenges of our world.

### References

- Anno, M. (1995). *Anno's seeds*. Paperstar Book (UK).
- Bråting, K., & Kilhamn, C. (2021). Exploring the intersection of algebraic and computational thinking. *Mathematical Thinking and Learning*, 23(2), 170-185.  
<https://doi.org/10.1080/10986065.2020.1779012>
- Brolley, L., Caron F., & Saint-Aubin, Y. (2018). Levels of Programming in Mathematical Research and University Mathematics Education, *International Journal of Research in Undergraduate Mathematics Education*, 4(1), 38-55 <https://doi.org/10.1007/s40753-017-0066-1>
- Buteau, C., Gueudet, G., Muller, E., Mgombelo, J., & Sacristán, A. I. (2020). University students turning computer programming into an instrument for 'authentic' mathematical work. *International Journal of Mathematical Education in Science and Technology*, 51(7), 1020-1041. <https://doi.org/10.1080/0020739X.2019.1648892>



- Caron, F. (2019). Approaches to investigating complex dynamical systems. In *Lines of inquiry in mathematical modelling research in education* (pp. 83-103). Springer, Cham.  
[https://link.springer.com/content/pdf/10.1007/978-3-030-14931-4\\_5](https://link.springer.com/content/pdf/10.1007/978-3-030-14931-4_5)
- diSessa, A. A. (2018). Computational literacy and “the big picture” concerning computers in mathematics education. *Mathematical thinking and learning*, 20(1), 3-31.  
<https://doi.org/10.1080/10986065.2018.1403544>
- Doerr, H. M. (1996). Stella ten years later: A review of the literature. *International Journal of Computers for Mathematical Learning*, 1, 201-224.
- Dohn, N. B. (2020). Students’ interest in Scratch coding in lower secondary mathematics. *British Journal of Educational Technology*, 51(1), 71-83. <https://doi.org/10.1111/bjet.12759>
- Gadanidis, G., Hughes, J. M., Namukasa, I., & Scucuglia, R. (2019). Computational modelling in elementary mathematics teacher education. In: *International Handbook of Mathematics Teacher Education: Volume 2* (pp. 197-222). Brill.
- Giabbanelli, P. J., & Mago, V. K. (2016). Teaching computational Modelling in the data science era. *Procedia Computer Science*, 80, 1968-1977.  
<https://doi.org/10.1016/j.procs.2016.05.517>
- Gonczy, A. L., Palosaari, C., Urban, N., & Mayer, A. (2022). From Simulation User to Creator: Helping Students See Inside the “Black Box” with Insight Maker. *The science teacher*, 89(3).
- Grover, S., & Pea, R. (2017). Computational thinking: A competency whose time has come. In S. Sentance, E. Barendsen, & C. Schulte (Eds.). *Computer science education: Perspectives on teaching and learning* (pp. 19–38). Bloomsbury Academic.  
<https://10.5040/9781350057142.ch-003>
- Lovric, M. (2018). Programming and Mathematics in an Upper-Level University Problem-Solving Course, *PRIMUS*, 28 (7), 683–698.  
<https://doi.org/10.1080/10511970.2017.1403524>
- Modeste, S. (2016). Impact of Informatics on Mathematics and Its Teaching. In: Gadducci, F., Tavosanis, M. (Eds). *History and Philosophy of Computing. HaPoC 2015. IFIP Advances in Information and Communication Technology*, vol 487. Springer, Cham.  
[https://doi.org/10.1007/978-3-319-47286-7\\_17](https://doi.org/10.1007/978-3-319-47286-7_17)
- Modeste, S. (2018). Relations between mathematics and computer science in the French secondary school: a developing curriculum. In: *ICMI STUDY 24*. <https://hal.archives-ouvertes.fr/hal-03184712/document>
- 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. <https://doi.org/10.1007/s10956-015-9581-5>



## **Coding, Computational Modelling & Equity in Mathematics Education Working Group C Equity, Diversity, and Inclusivity Report**

### *Working Group Leaders*

Diane Tepylo<sup>1</sup>, Annie Savard<sup>2</sup>, & Ricardo Scucuglia<sup>3</sup>

<sup>1</sup>Ontario Tech University; <sup>2</sup>McGill University; <sup>3</sup>Sao Paulo State University (UNESP)

### *Working Group Members*

Ghuseon Alqassab  
Marja Bertrand  
Gideon Christian  
Laura Hart  
Erica Huang  
Nadia Kennedy

Carolina Y.L.F. Graciolli  
Ami Mamolo  
Joyce Mgombelo  
Immaculate Namukasa  
Sheree Rodney  
Annie Savard

Ricardo Scucuglia R. da Silva  
Soheila Shahmohammadi  
Diane Tepylo  
Eleanor Smith  
Sultan Rana

### **Presentation of the Aims of the Working Group**

Coding and computational thinking (CT)<sup>1</sup> is being integrated into K-12 mathematics education around the world. Researchers have identified numerous potential benefits for integrating CT with mathematics including “make[s] abstract mathematical concepts concrete” (Wilensky, 1995, p.257), dynamic modelling to develop mathematical concepts and relationships (Gadanidis, 2015), support transfer of learning from the classroom to real-world settings (Lunce, 2006). Other researchers suggested: motivation to experiment; the development of mathematical intuitions; critical reflection; and working with abstraction and different representations (Howson & Kahane, 1986; King et al., 2001; Marshall & Buteau, 2014).

However, this rapid integration of CT & mathematics raises many potential equity issues, including equitable access to technology for coding, equitable access to quality instruction and instruction that makes coding attractive to a diverse range of students.

### **Equity, Diversity, and Inclusivity**

This working group addresses the issue of quality equitable pedagogy as the benefits of coding for mathematics learning are based on instruction that promotes problem-solving, computational thinking and connections to mathematical thinking for ALL students. Incorporating this equitable pedagogy is difficult for most teachers who have 1) little to no prior experience learning coding as problem-solving, 2) little access to quality teaching materials (Wu et al., 2020; Yadav et al., 2016) nor 3) no vision of integrating & connecting between CT / CM and mathematics teaching (Gleasant & Kim, 2018).

This working group explores equity issues around quality pedagogy for coding and mathematics, developing strategies and vision to make coding more inclusive. Participants will then examine existing coding and mathematics activities through an equity lens before modifying the activities to develop mathematical reasoning for all students. In this way the workshop hopes to develop equitable math and coding activities, and develop a vision for future adaptations. In creating more equitable coding and mathematics activities, and a framework for modifying other activities, this workshop hopes to harness “the power to change pedagogies and students’ experience of mathematics learning” (Ford, 2018, p. 27).

This working group aims to support the participants to:

1. Develop a vision of equity, diversity and inclusivity for coding/computational



Modelling education;

2. Improve teaching strategies to make coding more equitable, diverse, and inclusive;)
3. Build a framework to adapt coding activities to make them more inclusive.

### Developing Shared Understandings of EDI in Coding and Computational Modelling

Our first goal was to develop some shared understandings of EDI in the domain of coding and computational thinking. Our participants were from a wide range of backgrounds (schooling, subject matter, countries of origin), so we realized the need to share and develop participants' perceptions about EDI.

#### Equity, Diversity and Inclusion

Developing a vision of equity, diversity and inclusivity for coding/computational Modelling education means to present some definitions of these concepts. Equity is defined as fairness for all, especially about different identities such as “race, class, ethnicity, sex, beliefs and creeds, and proficiency in the dominant language” (Gutiérrez, 2002, p. 153). Weissglass (2002) raised important questions for educators about how racism, gender, sexual orientation, and culture affect student learning. He highlighted the role of the curriculum developer in engaging students and challenging inequalities. SSHRC website helped us to the following definitions<sup>1</sup>. In short, while equity refers to offer the same opportunities to each person (Auclair et al., 2022), diversity is defined as having people presenting different identities mentioned above, while inclusion is defined as everyone (diversity) being respected, values and supported.

Several graphics were shared with the working group to promote discussions around equity, diversity and inclusion. Figure one was cited as influential by participants. It contains two important additions to a oft-seen graphic: an image to represent Inclusion and written explanations of how each graphic illustrated the stated principle. Diane noted that these descriptions were important to learners: her pre-service teachers struggled to see equity and justice in the illustrations without support.

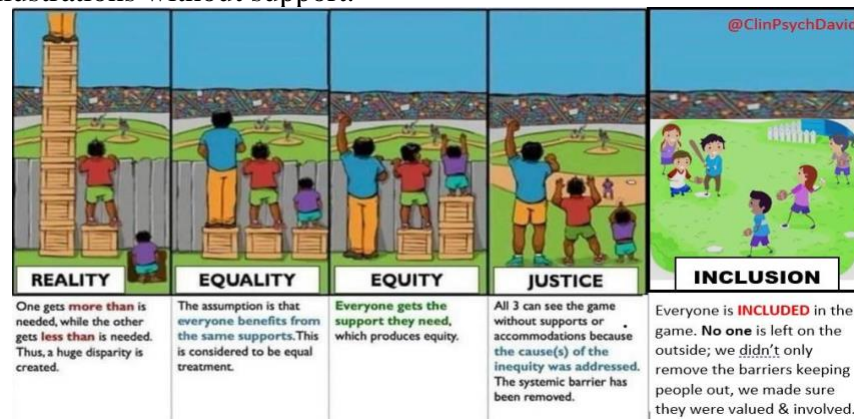


Figure 1. Graphic Illustration of Equity and Inclusion (Murphy, 2021).

We posed two tasks to participants to develop shared understandings. In small groups, participants discussed, “What lenses do we need to consider for teaching coding/computational Modelling equitably?” After a whole group discussion, participants were asked to consider examples and counterexamples of equity, diversity and inclusion in the coding/ computational

<sup>1</sup> [https://www.sshrc-crsh.gc.ca/funding-financement/apply-demande/guides/partnership\\_edi\\_guide-partenariats\\_guide\\_edi-eng.aspx#appendix-a](https://www.sshrc-crsh.gc.ca/funding-financement/apply-demande/guides/partnership_edi_guide-partenariats_guide_edi-eng.aspx#appendix-a)



thinking area. From these discussions, the following themes arose as EDI concerns within teaching coding and computational thinking:

**Equity** concerns that arose from the discussion were providing opportunities for all students to learn. Equitable access to technology and opportunities to learn were highlighted given the important learning and job possibilities that computational thinking provides (McCandless, 2018). Only providing coding instruction in summer camps and more affluent schools were areas of concern. Additionally, participants thought it was important to understand how artificial intelligence could be designed to perpetuate racism and stereotypes.

**Diversity** discussions centered around principles that all learners could “*see themselves in*” coding and computational thinking, remembering that individuals transcend categories of race, gender (intersectionality). Participants discussed bringing in culture, language, celebrations, but warned against superficial attempts such as making a Rangoli for Diwali. Diane shared that Scratch allowed students to code in multiple languages.

Related to equity and diversity is **inclusion**- ensuring that everyone is included and valued. Inclusion discussions included strategies to make the learning activities accessible to diverse ways of thinking (neurodiversity). Groups discussed implementing a variety of teaching strategies and representations. Creating learning communities where students support each in collaborative knowledge generation was stressed by several groups. Also important was allowing students choice: choice in tasks, and in strategies for approaching planning and problem solving.

The participants also discussed a variety of strategies to make all learners feel valued including tasks that are relevant to a variety of learners to encourage engagement and motivation. They stressed the importance of supporting students who are struggling and the importance of discussing the emotions that often accompany coding- even professional coders experience frustration and develop strategies to work through the frustration. Finally the participants stressed the importance of educators to build relationships with students.

### **Connecting EDI to the 5A’s**

After these initial explorations of equity, diversity and inclusion, we aimed to strengthen their connections to coding and computational thinking in mathematics education. We chose to use Gadanidis’s lens of the 5As- 5 affordances of CT (Figure Two).

In planning the working group, we struggled to find a lens that would help us analyze and work with BOTH EDI concerns and computational thinking. We decided on the 5As as a framework (Gadanidis et al., 2020 ) to connect computational thinking to Equity, Diversity and Inclusion issues. To that end, we considered the 5A models of affordances of computational thinking. The affordances are: Access, Agency, Abstraction, Automation, and Audience. In summary, Access refers not only to access to tools in socioeconomic terms, but also to the “low floor & high ceiling” design of tasks in which young students can be engaged with minimum prerequisite knowledge to investigate complex/rich mathematical ideas. Agency “allows students conceptual freedom to investigate ideas and concepts of interest” (p. 200). Abstraction regards fundamental characteristics and/or processes of concepts, emphasizing the emergency of tangible feel in coding. Automation, as a process that operates automatically related to algorithmization, highlights the dynamicity of coding and Modelling and brings up the joy of surprise and insight in mathematical activity. Audience refers to the possibility and relevance of sharing codes and models with others, who can re-use/re-mix them as well as improve the performance of codes and models.

Of note, Gadanidis and colleagues write extensively about the affordance of CT to support mathematical thinking- here represented by abstraction. Students can make a conjecture and then test it with code, receiving immediate feedback to whether their conjecture worked.

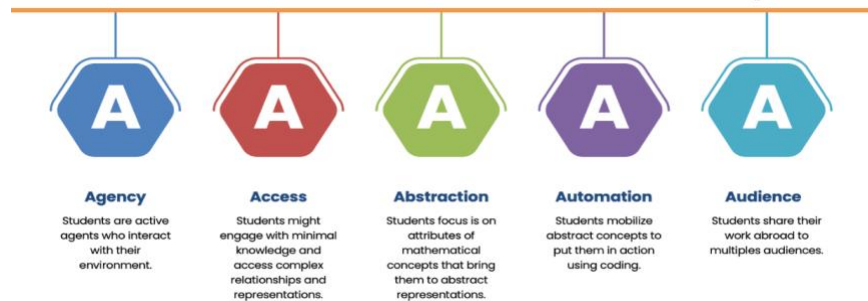


Figure 2. 5As: Five affordances of CT to Support Elementary Mathematics Education (Based on Gadanidis, 2017).

After a brief introduction of the five affordances, groups were asked to connect EDI concepts to one A on white boards for a change of modality. Groups embraced the challenge, using the 5A lens to categorize EDI concepts discussed previously, while raising new issues and developing practical strategies. Two groups graphics and subsequent discussions are discussed here to illustrate the nature of the small group discussions.

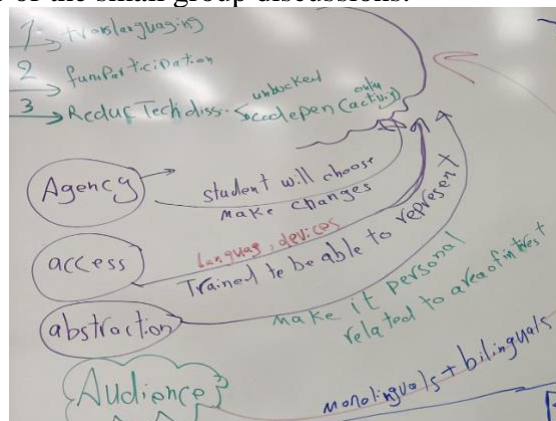


Figure 3. Connecting EDI and the 5 As - Group One.

For example, under audience and access group one - a group containing many English language learners and a teacher of immigrants, discussed the issues of translation and monolingual and bilingual audiences (Figure Three). They also discussed access in terms of access to technology and the importance of unplugged activities.

Group 2, with many experienced CT teachers, saw many connections between the 5As as indicated in Figure Four by dotted lines. This group detailed classroom considerations and strategies. For example, for accessibility, they identified the importance of language capabilities, background knowledge and various ways to demonstrate learning. They also noted the importance of access to start the activity (low floors). For abstraction, they stressed links to mathematical thinking including generalisation and rules. They also noted the value of coding for creating multiple representations of the mathematical concepts and immediate feedback on those ideas.

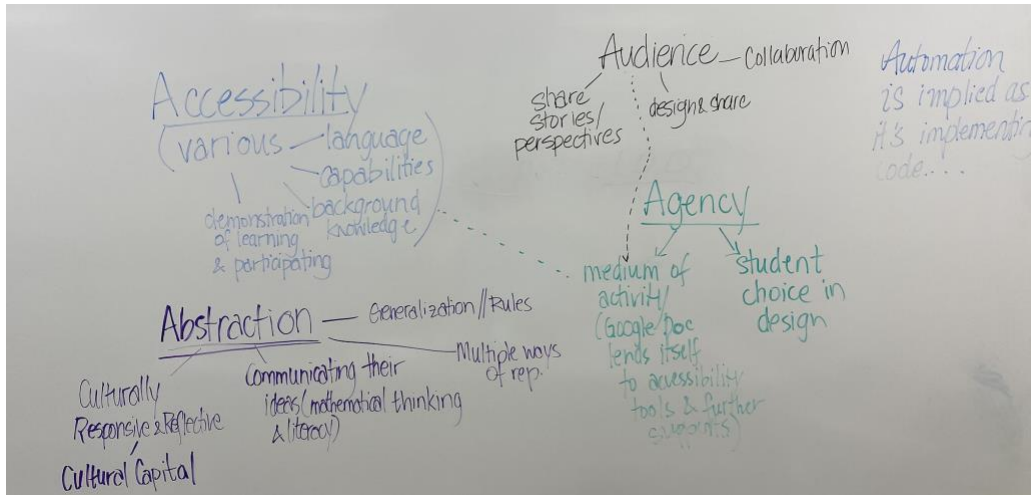


Figure 4. Connecting EDI and the 5 As - Group 2.

Through discussion around the 5As framework, participants made multiple connections to existing knowledge frameworks. Sheree Rodney understood agency through Pickering's (1995) perspective. He defines agency as the influence of one thing onto the other (students and technology). He believes that in performing a task the human (students) and the non-human (technology) are engaged in a back-and-forth interplay of resistance and accommodation which he called “the dance of agency”. He argues that learning takes place within this dance of agency.

This contrasts to Gadanidis’s agency that refers to students having control and choice (Gadanidis 2017, Gadanidis et al. 2019). Gadanidis and colleagues argue that this agency engages students and through that motivation leads to learning.

This in-depth exploration continued between workshop sessions and even after the symposium with participants continuing to explore how EDI considerations could push the affordances of learning CT. One participant, Sheree Rodney, continued consider how EDI connected with the 5As after the symposium sharing the following graphic with the facilitators. Sheree conceived EDI as the center with connections to the 5As. Under Equity & Automation, Sheree listed equitable access to digital resources which the first author connected to Gideon Christian’s keynote about many inequitable use of facial recognition (Buolamwini, 2017) ) and the use of banking apps where area codes are used to build racism (Beckles, 2021).

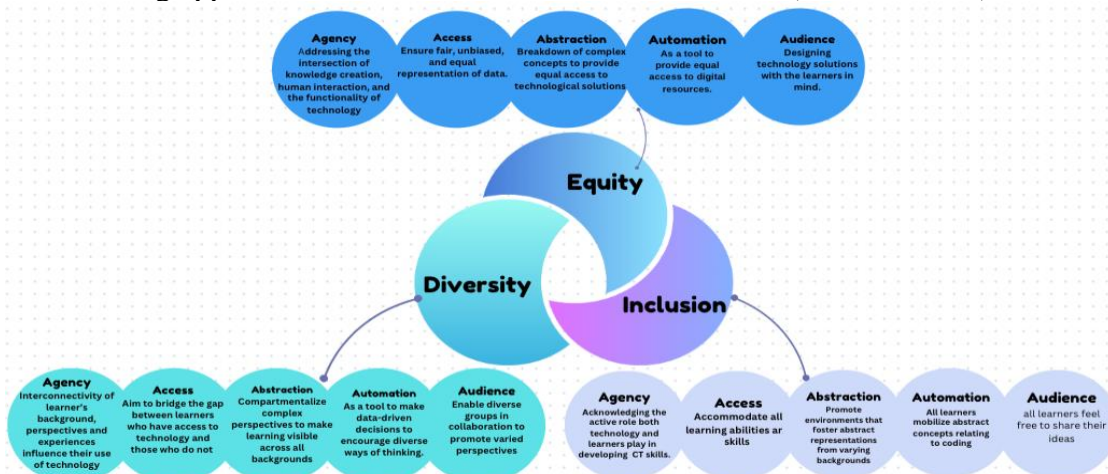


Figure 5. Ongoing exploration of the connections of The 5As and EDI - Sheree Rodney



This rich diagram is worthy of an extended discussion that is not possible in this proceeding. This need for extended conversations is characteristic of this working group. The discussions were productive, but participants noted they wanted to continue the discussions.

### The Task Adaptation Activity

On day two, the participants were tasked with applying their understanding of EDI issues in coding/computational thinking to adapting a task. [Some tasks were provided to seed the conversations](#) as were resources on AI, unplugged activities, less colonial views of coding.

Some groups worked with provided materials; others chose tasks from their experiences. In all cases, all participants adapted our general task to their own interests and perceptions. **NO PROCESS was imposed** as highlighted as an important characteristic in our EDI conversations..

One group chose to examine an activity from Erica’s class through the 5A & Equity lens. In this coding activity, students are scaffolded to draw a variety of polygons and spirals with lots of areas for choice and different processes. The group recorded their thoughts as comments in the google doc. As visible in Figure 6, the group noted the agency in choice and the access in low floors. Marja suggested diagnostic assessment to determine if more work is needed on mathematics vocabulary.

**Coding Activity: Spiral Polygons**

Today you will apply your geometry skills to make computer programs to draw various polygons and create your own spiral patterns. We will use Snap!, a block-based programming language.

**Think-Pair-Share: What does this code do?**

Look at each line, one by one.

- What does each command do?
- What is *steps*?
- Why 120 degrees?

Go to <https://snap.berkeley.edu/> and click *Run Snap!* now.

Copy the code above. You will need this code for Activity 1 and 2.

**Some Snap! Tips:**

- On the left is the *Palette* area (with colour-coded commands). The *scripting* area is in the middle (drag the blocks here to make your code), and the *stage* is on the right (see what your code does here).
- Whenever you want to delete a block, you can right click and select delete, or drag the code back to the palette.

**Useful script: Clear the screen**

The chat window on the right shows the following messages:

- Sultan Rana** (10:02 AM Yesterday): The whole medium of this task being on a Google Doc: AGENCY & ACCESS is provided through the ability of leveraging accessibility tools (G-Read & Write), translation tools (MS Translate), Glossary terms or multi-medium supports (YouTube tutorials, images, etc.); AUDIENCE (quick to share, show and collaborate through LMS's, shared docs)
- Marja Bertrand** (9:56 AM Yesterday): During the think-pair-share activity teacher can do formative assessment on their knowledge and understanding with respect to the math vocabulary. If they are having difficulty with math vocab, we can add some more scaffolding activities.
- Marja Bertrand** (10:02 AM Yesterday): Access: Start creating code with 3-4 steps so that students can have low-floor and then build up to more complicated code.
- Marja Bertrand** (10:00 AM Yesterday):

Figure 6. Analyzing a Scratch Task for the 5As and EDI considerations.

Another group examined an activity that Eleanor regularly uses with adults who have recently immigrated to Canada. Codepen is a Social Development Environment for creating web pages Social Development Environments are real-time collaborative programming tools with integrated social networking features.

Eleanor’s group discussed how this tool allows many of the 5As and EDI while allowing new immigrants to develop important skills. The site provides agency in terms of choice, and access in low floors and many supports. In creating websites, the creators have immediate audiences.

One of the ongoing Equity challenges with coding is access to technology for all learners. Unplugged activities are often promoted to develop CT where learners do not have regular access to computers or the internet.



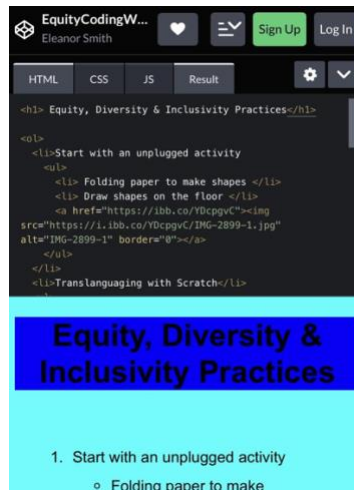


Figure 7. Analyzing a Codepen Task used with new immigrants.

However, many of these unplugged activities lack the engagement and immediate feedback that is associated with coding.

Carolina's group explored some paper folding activities that transformed the shape activity into seeds.

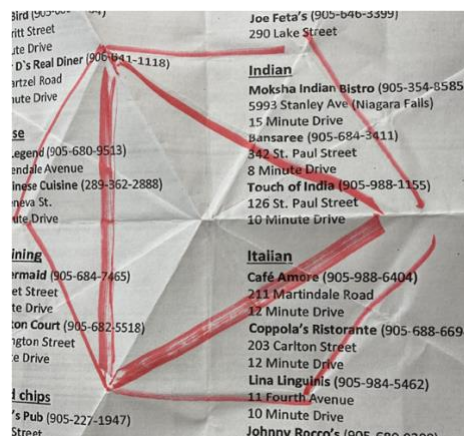


Figure 8. Example of a paper folding activity.

The purpose of the activity was to explore how we should fold and make a single cut in the paper to create regular polygons. Therefore, the idea was to work on the relationship between the central angle and the sides of the polygon, discussing mainly issues related to abstraction and patterns. In the image we can see an example that refers to the construction of a triangle and a hexagon, which in theory, when divided in half repeatedly, we have all polygons with 3, 6, 12, 24, 48, ... sides.

This disconnected activity impressed the larger group as it quickly provided visual feedback.

## Conclusion

In creating this working group, the facilitators aimed to create activities that not only discussed EDI in CT, but also integrated those ideas into all activities. With that focus on inclusive practices, we asked participants to co-prepare the report to the larger symposium.



Below we share the summaries of two groups that were indicative of the whole. One group summarized their key learnings and how the working group integrated these concepts throughout:

**Equity**

- Each person starts learning where they are based on their prior knowledge
- Each person is given the time and choices to proceed
- It is not open, but supported
- To choose language and to access students’ previous knowledge and backgrounds
- Choice is part of EDI versus telling them what to do

**Diversity**

- Brought together diverse interest and knowledge
- Activity itself allowed for diverse interest and knowledge
- Diversity in terms of
  - Demographics (eg. geographical region)
  - Ways of thinking

**Inclusion**

- All playing, not just watching
- All of us are participating and contributing in the activity in different ways
  - Gideon learning
  - Immaculate & Joyce working together
  - Laura supporting
  - Amy observing and reporting

Another group summarized their learning by examining the EDI considerations needed in planning, implementing, and assessing connecting to previous conversations about equity, Diversity and Inclusion in CT (Table 1).

*Table 1. EDI considerations needed in planning, implementing and assessing.*

<b>Planning the task</b>	<b>Implementing the task</b>	<b>Assessing/Building from the task</b>
Building who and where students are.	Enacting the task with EDI principles.	Taking from where the students are and want to go.
Reframing the task with EDI principles	Observing students and create conditions for agency in regard to EDI.	

Throughout the working group, participants were asked to share their insights and perspectives. Participants noted that the task connecting understandings of Equity, Diversity and Inclusion to Gadanidis’s (2015) 5A framework was very productive. This task provided a rich framework for discussion and extending our understandings of EDI when implementing coding and computational thinking into the classroom as evidenced in section 4 above. Participants noted that the activities helped them better understand the terms equity, diversity and inclusion:

“I Gained a better understanding of the differences between equity, diversity and inclusivity.”



“I found the extensions from EDI to Decolonization and Indigenization thought Provoking when applying the As from Gadanidis’ work”

Group members noted they “found this group was very respectful of each other views and a safe space to share your ideas. I learned so much from both the facilitators and the group members” and the need for continued learning. This is “Only the beginning...”

#### Footnotes:

Our group chose not to define the term Coding and computational thinking (CT) as CT was the conference theme and we wanted to focus our attention on Equity, Diversity and Inclusion.

#### References

- Auclair, I., St-Georges, J., Brière, S. et Keyser-Verreault, A. (2022). *Les biais inconscients dans un contexte d'équité, de diversité et d'inclusion*. Dans *Biais inconscients et comportements inclusifs dans les organisations*, Brière, S. et coll. (dirs.), pages : 7 -84. Presses de l'Université Laval.
- Beckles, O. (2021). Coding and Equity. Presentation at the Ontario Mathemaitcs Coordinators Association. <https://www.omca.website/>
- Buolamwini, J. (2017) How I'm fighting bias in algorithms. Youtube. [https://www.youtube.com/watch?v=UG\\_X\\_7g63rY&list=PLj62-wQeg\\_DhmYphxg70DhPEJcjfmnVET](https://www.youtube.com/watch?v=UG_X_7g63rY&list=PLj62-wQeg_DhmYphxg70DhPEJcjfmnVET)
- Ford, J. (2018). Digital technologies: Igniting or hindering curiosity in mathematics? *Australian Primary Mathematics Classroom*. The Australian Association of Mathematics Teachers, Inc. 23(4), 27-32.
- Gadanidis, G. (2015). Coding as a Trojan Horse for mathematics education reform. *Journal of Computers in Mathematics and Science Teaching*, 34(2), 155-173.
- Gadanidis, G. (2017). Five Affordances of Computational Thinking to support Elementary Mathematics Education. *Journal of Computers in Mathematics and Science Teaching*, 36(2), 143-151. Waynesville, NC USA: Association for the Advancement of Computing in Education (ACE). <https://www.learntechlib.org/primary/p/174346/>.
- Gadanidis, G., Hughes, J., Namukasa, I. & Scucuglia, R.. (2020). Computational Modelling in Elementary Mathematics Teacher Education. In: S. Llinares & O. Chapman (eds.), *International Handbook of Mathematics Teacher Education Volume 2: Tools and Processes in Mathematics Teacher Education*, 197-222. Koninklijke Brill NV: Leiden, The Netherlands. 10.1163/9789004418967\_008.
- Gleasant, C. & Kim, C (2018). *Use of blocked-based coding in teaching conceptual mathematics*. Paper presented at Association for Educational Communication and Technology Conference, Kansas City: MO.
- Gutierrez, R. (2002) Enabling the Practice of Mathematics Teachers in Context: Toward a New Equity Research Agenda, *Mathematical Thinking and Learning*, 4:2-3, 145-187, DOI: 10.1207/S15327833MTL04023\_4
- Howson, A.G., & Kahane, J.P. (eds). (1986). The influence of computers and informatics on mathematics and its teaching. *ICMI Study Series (Vol. 1)*. Cambridge, UK: Cambridge University Press.
- King, K., Hillel, J., & Artigue, M. (2001). Technology – A working group report. In D. Holton (ed.) *The Teaching and Learning of Mathematics at University Level: An ICMI Study*, (pp. 349-356). Dordrecht: Kluwer Academic Publishers.



- Lunce, L. M. (2006). Simulations: Bringing the benefits of situated learning to the traditional classroom. *Journal of Applied Educational Technology*, 3(1).
- Marshall, N. & Buteau, C. (2014). Learning mathematics by designing, programming, and investigating with interactive, dynamic computer-based objects. *International Journal of Technology in Mathematics Education*, 71(2), 49-64.
- McCandless, D. (2018) Diversity in tech: Employee breakdown of key technology companies. <https://informationisbeautiful.net/visualizations/diversity-in-tech/>.
- Murphy, D. (2021) Rework of the famous equality/equity/justice cartoon. <https://twitter.com/ClinPsychDavid/status/1407103431718969345/photo/1>
- Weissglass, J. (2002). Inequity in mathematics education: Questions for educators. *Mathematics Educator*, 12(2), 34–43.
- Wilensky, U. (1995). Paradox, programming, and learning probability: A case study in a connected mathematics framework. *The Journal of Mathematical Behavior*, 14 (2), 253–280.
- Wu, L., Looi, C.-K., Multisilta, J., How, M.-L., Choi, H., Hsu, T.-C., & Tuomi, P. (2020). Teacher’s Perceptions and Readiness to Teach Coding Skills: A Comparative Study Between Finland, Mainland China, Singapore, Taiwan, and South Korea. *The Asia-Pacific Education Researcher*, 29(1), 21–34. <https://doi.org/10.1007/s40299-019-00485-x>
- Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S. and Korb, J.T. (2014.). Computational Thinking in Elementary and Secondary Teacher Education. *ACM Transactions on Computing Education*, Vol. 14(1), 5: 1-16.



## On the Nature of Literacies and Literacy Development

Andrea A. diSessa  
University of California at Berkeley

My comments here will be drawn largely from directions instigated by Brent Davis's commentary on my talk. Those directions are plenty rich and important enough that I won't bother second-guessing in order to find "the optimal thing to discuss." I consider his comments to be offerings aimed to make computational literacy more comprehensible to a broader audience. Thanks, Brent.

In terms of "housekeeping," however, I would like first to mention that the paper Brent read and referred to was not the one my talk was based on. The conference organizers suggested that I make the paper that he read available—and it's one of my favorites. But I had already given a talk based on that paper to an overlapping audience. So, at the risk of some confusion, we made one paper available, but I talked about a somewhat different set of issues, from a different perspective. A paper more closely aligned with my talk is:

diSessa, A. A. (2016). Five powerful ideas about technology and education. In M. Riopel & Z. Smyrniou (Eds.), *New developments in science and technology education* (pp. 63 – 72). Heidelberg: Springer. Available at [https://link.springer.com/chapter/10.1007/978-3-319-22933-1\\_7](https://link.springer.com/chapter/10.1007/978-3-319-22933-1_7)

### The Nature of Computational Literacy

A lot of Brent's commentary addresses the issue of just what computational literacy can possibly mean. The fact that it seems mysterious to him is not at all beyond my comprehension or expectation. In the book that first introduced the idea in detail (diSessa, 2000), I start by explaining a bit about why it is unfamiliar and seemingly exotic. The short story is that it does not mesh with the widespread cultural memes of the day. So, in pursuing computational literacy, there is a lot of clutter<sup>2</sup> that needs to be moved out of the way, and a pathway to a new direction needs to be seeded and cultivated.

The cultural situation has gotten both a little better and a little worse in the 20 odd years since I published that book. What's gotten much better is the vivid public presence of the meme (culturally wide-spread idea) that programming is something that every student should participate in. When I got started in technology and education, a then-current predominant meme was the reverse: Nobody needs to understand programming any more than they need to understand how their washing machine works (Really! References on request.). What has gotten worse is that the conception of what programming gives to students has been muddled and misdirected (my judgement) by alternative narratives that have become popular. Chief among them is the idea that people just get smarter by engaging in programming, in whatever field they choose to pursue. Other current memes include "you can get a good job in the tech industry this way," and "this is for the good of society; knowledge of what's going on under the hood affords a kind of public oversight to avoid the mayhem that bad-actor technologists can inflict on society." This last civics-oriented meme favors understanding something of the insides of things like algorithms that make recommendations, or order search hits for individuals—or soon, if not already, the perils of AI in the public forum.

---

<sup>2</sup> Brent's search of the internet for wisdom on what computational literacy means, I'm sorry to say, documents a lot of the cultural "clutter" that mostly obscures better-founded ideas.





I have always thought, and continue to think, that people can self-bootstrap into the concept of computational literacy via their massive experience with textual literacy. It simply is not difficult to see the depth and breadth of effects of literacy on our civilization. Imagine: No “nation of laws, not men”; no history; no scientific paper; no textbooks; no classics of literature. Every modern country is measured by its literacy rate, and national programs to improve literacy are strong, even in countries with high literacy rates, like the U.S. and Canada.

I have also found it easy to bootstrap conversations about whether, for example, algebra/calculus is a literacy, or not. People do understand the power and some of the main characteristics of textual literacy, and against that, they make judgments about whether knowing algebra/calculus meshes. Now, there are many ways to slice this, given that literacy is not a technical term (with attendant clarity of breadth and meaning). Not *everybody* needs algebra/calculus, and even those who do need it don’t need it every day, for many different purposes, unlike textual literacy. Algebra/calculus does meet my own technical definition of literacy. (Consult the discussion on “what is a literacy?” in the first chapter of diSessa, 2000.) But, I can definitely see the sensibility in people drawing lines in a different place.

So, here’s a core observation. Literacies can come in many, many varieties, at least as different as the many communities that adopt literate practices at their core. What we want to do is not only—or even primarily—decide whether we want to call something a literacy, but instead establish the modes and importances that any literacy-like set of practices entails.<sup>3</sup>

In that spirit, I’d like to lay out some plausible futures in the development of what I’d call computational literacy, suspending any certainty in using the term, opening space for readers to make their own decision. Literacy development is a project of immense uncertainty, which is why I often accede to the observation that “we just don’t know what (future) computational literacy will look like.” But I simultaneously maintain that there is no doubt that literacy-like things will (or, even more certainly, *can*) come to exist. I’m sorry that we cannot foresee their particular forms more clearly. But such is the fact of diversity in literacy types, and their complex relations to knowledge and social structures.

All of what I say below involves fairly simple extrapolations of things we (myself and my extended community) have experienced already in teaching actual students in actual classrooms, as laid out in my talk. So, this is not pure conjecture, but it is extrapolating what we have actually seen with students and teachers to the level of (potential) societal literacy.

### Multi-Causal Complex Systems

Before I get going with the nitty gritty, I want to take a minimal stab at characterizing, at a very general level, the nature of literacy, to compare with potential new literacies. I believe literacy is a very complex, *multi-causal* system. “Multi-causal” is important in that the system contains sub-systems with very different properties. Global warming is multi-causal because it depends not only on the dynamics of the atmosphere, but also on human reactions to relevant facts, and even political responsibility and action.

First, and most obviously, there is the technical perspective, which concerns the computational medium, the material substrate (like text) for the literacy. I think it is obvious that a substrate for a true literacy has to have properties that one does not typically measure for things like “programming languages.” The first focus I would attend to is simplicity and learnability,

---

<sup>3</sup> One thing that has gone wrong in discussing things like computational literacies (or computational thinking) is that the focus has been too much on computation, and not enough on literacy practices, the bread and butter of how familiarity with a representational system (like text, or programming) affects community-wide intellectual practices.



over broad age-ranges, for explicitly educational tasks. So, I will assert with great confidence (although, of course, I accept some people will contest it), that the system we have been using, the Boxer computational medium, is far more learnable than programming languages like Python, which are (by default) by and large aimed at already programming-literate folks. We designed Boxer explicitly to be significantly easier to start programming in than Logo, the benchmark I “grew up” with, just entering the technology and education field. I think we succeeded admirably (diSessa, 1991). In fact, early results working with teachers and students were so positive that we just gave up on a project to prove learnability, in favor of simply starting to teach with it.<sup>4</sup>

A second technological dimension is expressiveness. This relates directly to the remediation idea in my five powerful ideas exposition. How does the medium make it easier to express and explore ideas from a wide range of arenas, such as much or most of traditional school mathematics or science? This is transparently important for the goal of computational literacy, but it is simply not a visible goal of “garden variety” programming languages. Scratch most definitely was designed with learnability in mind, but my own judgement is that it has a pretty hard limit in terms of expressiveness.<sup>5</sup> I will just note, again, that expressiveness (for non-technical users) has very little, if any, presence in the design of most programming languages, and considering the issue brings us into the cognitive worlds of students in relation to an analysis of subject matter. How well does any computational system do at expressing the idea of “force” or “prime number”? Expressiveness is also simply not the same thing as comprehensibility of computational systems. So we see diverse and very different causalities coming together in designing and/or evaluating technology for a computational literacy.

A very different causality (very different science) undergirds the social/cultural dimension of literacy and its development. How do we understand the various subcommunities that might adopt a new literacy, their characteristic goals and capacity to change from existing practices? In order to consider the cultural dimension, one would not even be tempted to consult the same people as for the technical dimensions. Academia is famously “siloeed,” definitively separated, especially as it relates to computational literacies. The bridging that is necessary in considering new-literacy development needs to happen elsewhere than inside existing academic disciplines.

I’ll have to leave the topic of multi-causal science now. Mostly, I just want readers to have a rough sense of what is involved, and then to be in awe of how complicated the issues are. We have miles to go to get the relevant sciences to speak clearly, in concert with each other, to optimally design for and evaluate new-literacy initiatives.<sup>6</sup>

---

<sup>4</sup> There were some technical and careful published results, even from outside the Boxer community. Results of a study done by scientists at IBM Germany about constructing complex data objects were so stunning that there was notable public pushback, which I think was safely be dismissed as a kind of “sour grapes.” Some of our own results with sixth graders involved topics like scoping, which is seldom even discussed with novice programmers.

<sup>5</sup> Obviously, like many other elements of my argument, here, this deserves elaboration, which I won’t have space to explore here.

<sup>6</sup> I think in the short term—or possibly even in the long term—we will need *people with good intuitions* about many/all of the diverse collection of causalities involved in new literacies, while the different relevant sciences get their acts together to address literacies and their development adequately. You might think that the theory of literacy would be a good place to look for a unified treatment of the multiple causalities involved. Sadly, this is not in the offing. Looking through the history of literacy theory, early and mostly purely cognitive approaches have said virtually nothing about the sociocultural side of things. Newer, sociocultural approaches seem to have completely rejected any cognitive insights, such as re-representation. And, the design of computational media, meeting the demands of new literacies, is such an exotic enterprise that I would be daunted to try to build a decent reading list.

What I'd like to do now is to discuss how computational literacy will or won't look like textual literacy, which is the prototype reference model. As I mentioned, while it is speculative, this discussion is also grounded in the many experiences we've had using and teaching with a computational medium. I'm going to proceed by examining the repertoire of "objects of literacy" that are parts of textual literacy and/or will be part of computational literacies.

### **"Objects of Literacy": Size and Patterns of Re-Use**

I want first to consider the size of *objects of literacy*, and, more generally, the distribution of them across the spectrum of sizes. Maybe the prototype for literacy—and arguably the most important class of objects involved with textual literacy—is books. These are pretty large, as scales of literacy objects go. But, when we think of "great literature," especially that which has been most influential to civilization, books most definitely come to mind.

Thinking across a span of sizes, "aphorisms" and "quotes" occupy the smaller end of the textual literacy objects spectrum. These are broadly apparent but not nearly as important as book-level literacy objects.

Will there be book-sized objects of literacy in computational literacies? In fact, the image of a book, augmented internally by computational objects—such as simulations and microworlds (learning/exploring environments targeted at particular topics) was one of the guiding images for the creation of Boxer.<sup>7</sup> And it's an image that I have found resonates with many people. But, here's a cautionary tale about where we are in the development of computational literacy: I am now writing a physics textbook that would absolutely work marvelously as a Boxer book. However, I feel I simply can't risk all the ideas I aim to put into this book just falling by the wayside of technology change. Textual books have lasted centuries. Computational books, right now, would have a very uncertain future based just on the technology available to support them. So, assuming computational literacies develop, "books" may, ironically (in view of their importance) come late in the game.

Difficulty in developing long timescale, book-sized contributions is one reason that smaller-sized objects of literacy may predominate, at least early in the development trajectory. But, I think the prevalence of smaller-sized objects gets a more general and important boost in computational literacies. Quotes and aphorisms have their place, but little tools and computational objects of various sorts have proven far more durable in their use than corresponding textual objects. I think the key is that those small things *do* things and can be used in a lot of different environments with other computational (and textual) objects. Quotes and aphorisms, to take the same "small/textual objects" examples, tend to be very specific, and are not malleable in the way that computational objects can be. Those who saw my lecture should have noticed a bunch of standard little tools—like buttons, "clickers" (to turn things on and off), sliders, even vectors—that can be cut and pasted into (and easily computationally integrated with) any "new" microworld or work environment. Color pallets provide a good "transfer" example. They were developed for an image-processing toolset but could then be just cut and pasted into the Chartworld environment to allow students to pursue aesthetic aspects of number patterns. See also diSessa (1997) for more examples and an elaboration of the "flexible, re-usable computational objects" idea. You can write me to get a copy of that paper.

I make here just one more observation about the size of (computational) literacy objects. We have found that simple versions of tools and other objects are inordinately powerful. First,

---

<sup>7</sup> Boxer's core organization is a hypertext processor—text augmented by hierarchical structure (boxes inside boxes) and links across them, which are called "ports" in Boxer.

they can do an important job quite directly in many contexts, while being simple enough to be easily understood. And, since they are easily inspectable and modifiable (prime goals of Boxer), they can be simply adapted in small doses to work better in different use-cases. Contrasts are both (1) “apps” (e.g., iPhone apps) that are almost completely closed to modification or for re-use in different-than-originally-intended contexts, and (2) large *applications* such as Microsoft Word, which, in combining resources for every imaginable use, become overly complex and difficult to use. In Word, I always have trouble finding certain resources I need for various purposes, even though I know they are available—somewhere—in the application.

To summarize, computational objects of literacy will appear at every scale, similar to conventional literacy. However, I predict a strong “tilt” toward smaller scales, because of the ease of sensible modification and re-use in different contexts, in conjunction with other small (or larger!) computational objects. The long path to a true literacy is implicated in the cautionary tale that some of the most powerful objects of literacy (book-sized objects) are hampered during this, the early phases of literacy development, by comprehensible but potentially catastrophic “accidents” (such as a shifting infrastructure for computational media).

### A Manifesto?

Brent made a point of asking me whether the paper he read was a “manifesto.” Or, possibly, he was asking whether the very idea of “computational literacy” *constitutes* the core of a manifesto. Without pursuing his actual intentions, I’d like to provide some clarity.

The first two “hits” in a Google search resulted in the following: A manifesto is: “... a written statement of a person or group’s beliefs, aims, and policies, especially their political beliefs”; “... a call-to-action that works to ignite the masses.” A couple of edits adapt the definition to the present case. First, computational literacies are not political in the common meaning of that term. At the top level, I don’t perceive a difference between the political right or left with regard to whether people should learn to read and write. (Of course, the *content* of what’s written about is a different matter, but not obviously related to learning the medium.) “Policies” might be a little questionable in the context of literacies, too, given its normal association with politics, which I’ve already jettisoned.

The second edit is about “igniting *the masses*.” A more sensible adaptation, in this case, is “*everyone*, especially those interested and capable of moving the agenda forward.”

These aside, I’m happy to accept the claim that I am often advocating for computational literacies and urging action. In that sense, some of my writings may be properly regarded as manifestos.

On the other hand, I think it is very important to keep clear on the fact that “manifesto” is a rhetorical form that has very little to do with the nature of the thing that is being advocated. I think it warrants extra care to keep the default political context for “manifesto” at bay. The nature of the thing I am advocating is quite distinct from that for which manifesto is usually invoked. I would say, at least, that the object of my advocacy is:

**Scientifically accountable:** Even though the sciences of the various causalities that are involved with literacy are not sufficiently developed or integrated to be definitive, I most definitely have scientific accountability in my sights. My community and I have written papers about: (1) how to understand and evaluate comprehensibility of computational system (e.g., diSessa, 1991), (2) how computational representations actually work, in the moment-by-moment thinking of students, to develop scientific concepts (e.g., Parnafes, 2007), (3) how these representations can

evoke relevant student intuitive knowledge, to support their learning productively. In all these pursuits, I am open to the usual scientific scrutiny, evaluation, and dispute.

**Practically accountable:** In the end, this whole enterprise, in my estimation, comes down to “can we substantially improve student learning of science and mathematics in the mode of computationally literate practices?” Many of my own (and those of my extended research group) published results are various versions of “look how much earlier we can teach X than is presently true,” or “look how we can teach important things that have never been very teachable with non-computational media” (see diSessa, 2008), or “look how we can make the learning process much more enjoyable and engaging for students, without vitiating the ultimate goal of having them come to a deep understanding of science and mathematics.”

To counter the (hidden but perhaps insidious) influence of the standard political use-context of “manifesto,” and also to be truer to the nature of computational literacies as an object of advocacy, I might prefer the term “programm” (from the German, or as sometimes rendered “program” or “programme”). My personal favorite example from mathematics is Klein’s Erlanger Programm. In his monumental 1872 paper, Klein advocated a large and long-term agenda of integrating geometry and group theory. His paper was regarded by some as rather “philosophical” rather than typically mathematical. But the effect of that paper was, over the long haul, exactly what Klein was advocating.

In 1872, the E.P. [Erlanger Programm] was 20 years ahead of its time; it would take at least that long for the new perspectives of Klein and Lie to gain general acceptance.

However, their ideas would take deep root; in fifty years it would become commonplace [to fully embrace Klein’s programme]. (Birkhoff & Bennett, 1988, p. 152). [With minor edits and clarifications of the original quote]

At 20 years, computational literacy might be a little behind the standard that Klein set. But, given 50 years, perhaps advocates of computational literacy, as a programme, can catch up. That is well within the time-scale of what I have taken to be a sensible expectation for really developing a broad computational literacy: 100 years (given precedent for historical development of literacies, specifically both mass literacy with text, and with algebra/calculus).

### Avoiding Getting “Winged”

Brent asked the basic question of how we can look past the noise of contemporary buzzwords and transient obsessions in education to get to the crux of profound, long-term progress. My somewhat facile answer is, of course, to attend to the idea of computational literacy. This larger frame, valuable especially for literacy skeptics, allows us to be critical and look for robust science at the base of any possible direction, and to look to people who have a commitment toward actually changing teaching and learning with technology, people who present us with innovations and good documentation of “what happened, how that happened, and in what way is it better.”

Jeannette Wing’s “computational thinking” is my constant reference for “currently popular things to avoid.” So, I propose to run through a quick-and-dirty evaluation of Wing’s conception of computational thinking, in direct comparison to each of the five powerful ideas that I proposed as worthy directions in my talk. I add sample references to the work of my community in cases that previous citations in this article have not touched.

1. *The very notion of computational media and new (computational) literacy:* There is no even minimal reflection of anything that is identifiably in this direction in Wing’s writings. There is no mention, for example, of the critical task of developing a





*literature* for new media. Indeed, there is no mention of learning any particular science or mathematics, per se. Instead, quite clearly, Wing invoked the questionable—and I think discredited—idea that special disciplines (like computer science, or, historically and famously, Latin or mathematics) uniquely develop general powers, such as abstraction, that make performance in every disciplinary context better.

2. *Re-mediation*: Computational literacy, in my view, inherently draws on the idea that computational representations can be more apt than many others for particular arenas of science and mathematics. It cannot be universally optimal; every representational system has strengths and weaknesses or blind spots. But my claim is that computation is both easier to learn and far more applicable to a diverse range of scientific and mathematical things-to-learn than representational systems such as graphs or even algebra/calculus. A lot of the enterprise is to find the strengths of computational representations, capitalize on them, and to find other ways to improve instruction when computation alone is insufficient, or, indeed, not directly helpful at all. Wing’s sole bridge to particular subject matter is via skills that are simply not particular to any subject. They are, by claim, domain neutral.
3. *Engagement and Activity Structures*: Wing has explicitly advocated making computer science instruction more fun and engaging. But she has nothing to say about making specific science or mathematics similarly more engaging. As far as I am aware, there is precious little literature on programming, per se, making the learning of anything else more engaging. In our work, we have not only sought engagement as a goal, but researched the ways in which some topics or engagement strategies work well...or poorly. (Sherin et al., 2005).
4. *Open Toolsets*: Wing does not treat in any way how programming, as it exists, is limited in advancing educational goals. Open toolsets, in contrast, is a partial architecture for learning objects that has special advantages over “just programming.” These include adapting quickly to particular needs (such as different subject matter), and the ability to easily co-opt resources from one educational application into another.
5. *The LaDDER model*: As far as I have read, Wing never explicitly mentions culture or social elements of developing the competencies she aims for. The whole sociocultural layer of causality is, thus, missing. The LaDDER model is, in contrast, a social architecture for overcoming cultural divides and suboptimal concerted work that we observed in real-world empirical data about educational software development. It thus represents the core, larger observation that a new literacy is, at its heart, an accomplishment of a culture or society. It is not simply a thing in any person’s head, conveyed by programming. diSessa, Azevedo, and Parnafes (2004) provides a slightly extended and situated discussion of the LaDDER model and the problems it addresses.

Beyond this “five-ideas” list, I am unaware of any scientific literature showing that Wing’s program actually works to make people “smarter” or passes any test of scientific legitimacy.

### **Forward!**

To close on a positive note, I repeat my response to Brent when he asked for guidance on which of the very many paths, implicit or explicit, in my five powerful ideas talk should be pursued urgently, now. For myself, I have chosen to explore new things to teach (radical changes in the very curriculum of schooling), and simultaneous improvement in student engagement.



These are the most exciting and important possibilities I see, and I have a special affinity (and maybe I've developed not-so-common skills) for pursuing them. But there's such a broad front that is open to us that I feel no particular need to proselytize for my own interests and preferred directions. We'll all benefit from diverse engagements in the larger program.

### References

- Birkhoff, G. & Bennett, M. K. (1988). Felix Klein and his Erlanger Program. In W. Aspray & P. Kitcher (Eds.), *History and philosophy of modern mathematics* (pp. 145-176). Minneapolis: University of Minnesota Press.
- diSessa, A. A. (1991). Local sciences: Viewing the design of human-computer systems as cognitive science. In J. M. Carroll (Ed.), *Designing Interaction: Psychology at the Human-Computer Interface* (pp. 162-202). NY: Cambridge University Press.
- diSessa, A. A. (1997). Open toolsets: New ends and new means in learning mathematics and science with computers. In E. Pehkonen (Ed.), *Proceedings of the 21st Conference of the International Group for the Psychology of Mathematics Education*, Vol. 1 (pp. 47-62). Lahti, Finland.
- diSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge, MA: MIT Press.
- diSessa, A. A. (2008). Can students re-invent fundamental scientific principles?: Evaluating the promise of new-media literacies. In T. Willoughby, & E. Wood (Eds.), *Children's learning in a digital world* (pp. 218-248). Oxford, UK: Blackwell Publishing.
- diSessa, A. A., Azevedo, F., & Parnafes, O. (2004). Issues in component computing: A synthetic review. *Interactive Learning Environments*, 12(1-2), 109-159.
- Parnafes, O. (2007). What does "fast" mean? Understanding the physical world through computational representations. *Journal of the Learning Sciences*, 16(3), 415-450.
- Sherin, B., Azevedo, F., & diSessa, A. (2005). Exploration zones: A framework for describing the emergent structure of learning activities. In Rosebery, A., Warren, B., Nemirovsky, R., & Solomon, J. (Eds.). *Everyday matters in science and mathematics* (pp. 329-366). Wahwah, NJ: Lawrence Erlbaum Associates.



## Some Thoughts on Andrea diSessa's Five Powerful Ideas about Technology and Education

Brent Davis  
University of Calgary

Let me begin by offering a subjective history of the influence of technological innovation on school mathematics over the last half-century, interpreting those 50 years of evolution through the lenses of a grade-school learner, a mathematics undergrad, a secondary-school teacher, a graduate student, and a university-based researcher.

And let me structure this account around a learning device gifted to me by my mother. She loved mathematics and instilled in me a habit of asking “WHY ...?”—or, in long form, “What happens when you ...?” ... add a number that’s 1 greater ... extend the idea to infinity ... draw a parallel line that passes through the origin ... substitute with a friendlier term .... The device continues to serve me well. On to my history:

I started high school in the mid-1970s, just after two older siblings had graduated. I had tracked their high-school studies with great interest, and I arrived at the door of the school eager to finally make use of my slide rule and personal copy of *Knott's Four-Figure Mathematical Tables*. No part of my siblings' studies had captured my attention more than these artifacts, each a nexus of so much insight into number and number operations. I have no idea how much time I'd spent exploring WHY's with them in the years before reaching high school, but it was enough to prompt my parents to express concern at my obsession.

But, alas, the slide rule was not to be part of my official school mathematics experience, and *Knott's Tables* ceased to be used at the end of Grade 10. That's because hand-held calculators entered the high-school mathematics classroom at about the same time I did, and they were with me through my undergraduate experience. While I always thought these machines were a bit of a step backward, since they seemed to conceal the relationships and possibilities that slide rules and function tables so openly present, I appreciated their expansive new spaces of WHY's. Being able to engage with sequences of rapid calculations pulled at my imagination. I recall vividly, for example, the first time I really “felt” exponentiation, as repeatedly pressed the “=” button after entering “ $2 \times 2$ .” It was exhilarating.

When I returned as a teacher to the secondary school classroom in the early 1980s, graphing calculators were all the rage. They enabled an impressive new class of WHY's—ones that relied on many and varied simultaneous calculations rather than sequences of singular processes. Moreover, they communicated their results in both numbers and images.

By the late-1990s, those capabilities had been elaborated into tools that enabled WHY questions that involve millions of movement-and-interaction-generating calculations on a computer screen, affording access to mathematical concepts and abstract insights that were just not available to me or my students only years earlier.

I could go on. But I'll pause here to wonder about diSessa's assertion (during his presentation) that “*the computer is a once-in-several-centuries innovation.*” I suppose that's true, but I can't resist rearranging the words to assert that *the computer is several centuries of innovation at once*. Illustrated on the near-trivial level the experiences noted above, the computer incorporates-but-transcends all the technologies that framed and enabled my learning. All at once, the computer invites every type of WHY that I've been able to ask—which is to say, the computer feels to me more like an iteration than an innovation.

## What, Exactly, is “Computational Literacy”?

Why does that detail matter to me?

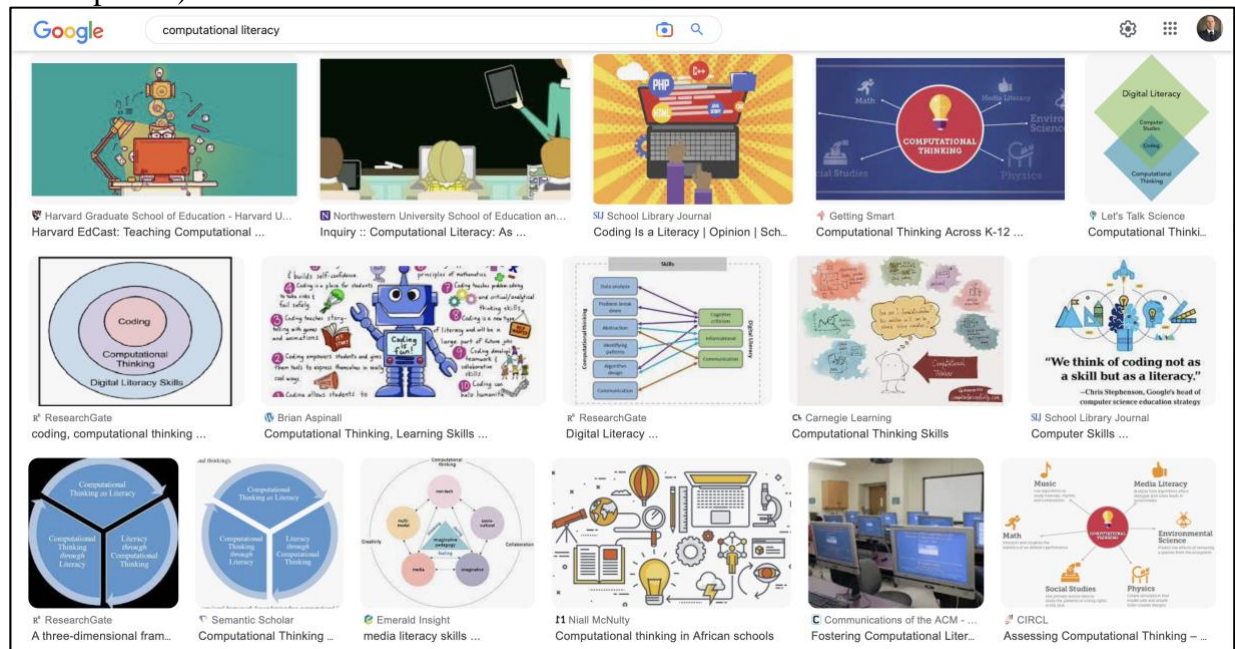
Well, it has everything to do with a personal struggle to make sense of what diSessa might mean by “computational literacy.” In preparation for this response, I took multiple runs at understanding the construct (in addition to reading diSessa’s 2018 article). I’ll speak to three of them, starting with information provided by Wikipedia (on 2023 April 26):

**Computational literacy** is a term that is used to describe the broad ability to apply computational thinking and awareness of the range, scope and limitations of computational techniques. It is distinct from the ability to perform math calculations by hand, instead anticipating that the actual computations will be performed by computers.

A leading advocate of Computational literacy, Conrad Wolfram explains the difference as “The maths taught around the world today does not fit how it is used in the real world. Computation technology is more accessible than ever before, but no curriculum in the world assumes it exists. Instead, it is focussed on the mechanics of hand calculation, rather than the essence of real-world maths.” Andrea diSessa has stated that “computers might fundamentally change learning.”

Of course, it was reassuring to encounter diSessa’s voice here. But I honestly couldn’t extract much insight into computational literacy from this bit of text—other than to infer that we educators seem to be doing a lousy job at nurturing it. What I gleaned from Wikipedia, then, is that there’s a tendency to define computational literacy in terms of something that modern schools aren’t addressing adequately. Fair enough. Schools have always been more inclined to look backward than forward. They’re notorious for focusing on literacies that are fitted to previous versions of society.

Hoping for a more broad-based glimpse into the construct, I turned to a favorite strategy of using an image search. This tactic rests on a no-doubt-fatally-flawed assumption that such searches can operate as cultural inkblot-tests—glimpses into what is operating in the collective unconscious, just beneath the explicit articulation. Here’s what Google Image brought back (on 2023 April 26):







What might one pull from this? I see three main themes (and I acknowledge I'm likely missing some other obvious ones): Firstly, several images appear to confirm my initial impulse, reflected above, to index computational literacy to recent evolutions of digital technologies. Secondly, in several others, there seems to be a conflation of "computational literacy" and "computational thinking." And, finally, a couple of the images might be taken to suggest that computational literacy arises in the meeting of various disciplines and/or other culturally privileged literacies (e.g., mathematics, music, and social studies). But, of course, none of that is especially surprising. Or, for that matter, enlightening. This image-search strategy usually affords some sort of unexpected insight for me, but in this case the results seem to served only as further confirmation that computational literacy isn't a broadly or readily understood notion.

That took me to my third strategy, which was ask ChatGPT. I agree with Noam Chomsky's assessment of this technology, as little more than automated and unabashed plagiarism, as so I anticipated its result would be a text version of my image search. Here's how it responded to the question, "What's 'computational literacy?'" (on 2023 April 26):

Computational literacy refers to the ability to understand and use computers and computational tools effectively. It involves not just basic computer skills such as operating systems and software, but also the ability to think critically about the ways in which computing technology shapes our world, and to use computation to solve complex problems.

At its core, computational literacy is about being able to read, write, and think in a world that is increasingly shaped by digital technology. This includes being able to use algorithms and code to manipulate data, as well as to critically evaluate the results of those manipulations.

While still not completely satisfying, this description felt a bit meatier than I was expecting, especially that mention of "the ability to think critically" that's tucked amid the otherwise utilitarian elements.

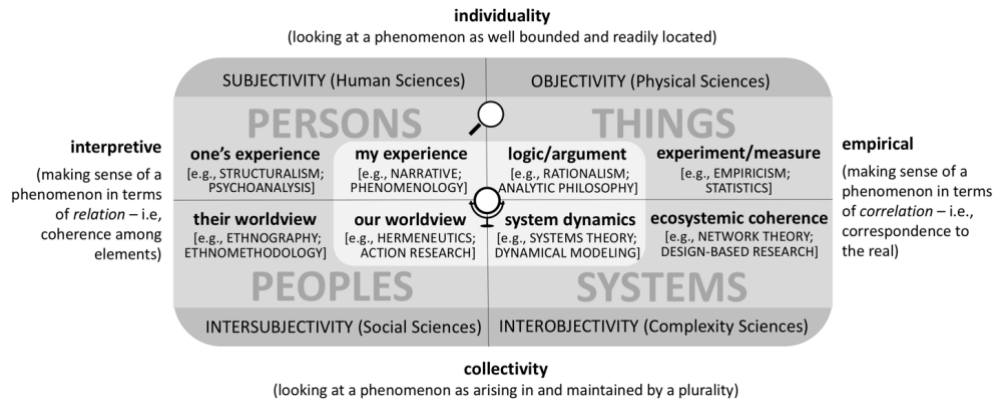
I do have a point in all this. Cutting to it, if the construct of computational literacy is to have any cultural value, it seems to me that what is intended by the phrase must be readily accessible. And I confess that I don't quite get it, even after having paid close attention to multiple of diSessa's explications. Despite those, I find myself unable to reassemble a description that doesn't devolve into the word (or graphic) salads presented above. My default continues to be an extrapolation from my understanding of "mathematical literacy—which, in a nutshell, I understand as having to do with a fluidity of movement through a mathematized (computationalized) world ... not just using/applying mathematical (computer-specific) knowledge, but recognizing, analyzing, evaluating, and/or extending, as appropriate in an interplay of not-necessarily-conscious knowing and doing.

### **What, Exactly, is Being Presented?**

Let me edescribe my struggle in understanding in another way.

I'm in my 30<sup>th</sup> year of employment in the academic world, and one of the consequences of being around that long is that I've taught a good number of research methodology courses. The graphic below is a coarse representation of how I organize those courses. I'll defer detailed explication to another piece of writing (Davis, 2019), but I will mention here that my strong impression is that, if I wanted to research computational literacy, I'd probably be framing things in terms of phenomena and methodologies landing on the left side of the image.

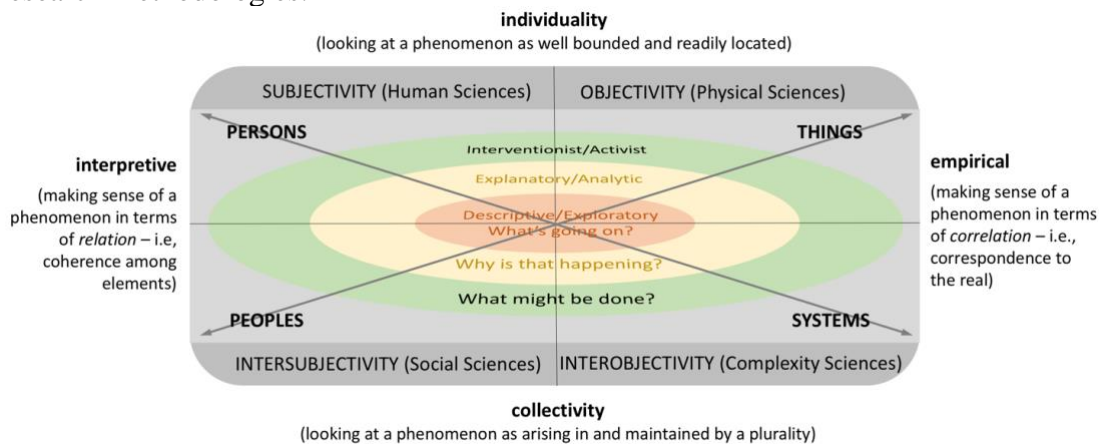




But, yet again, that doesn’t seem to tell me much. So, I’ll move to the graphic below, which is intended as a sort of overlay to the one above. The nested regions in this image highlight three distinct types of research questions and researcher intentions:

- Explorative research that leads to descriptive reporting,
- Analytical research that generates explanatory commentaries,
- Activist research that are often presented interventions and manifestos.

Such intentions and question types, I believe, operate across most educational phenomena and most research methodologies.



I illustrate these three attitudes as nested because, in my experience, reports (i.e., descriptions of explorations) can stand alone in the published literature, commentaries (i.e., explanations and analyses) typically begin with reports but extend them, and manifestos (i.e., interventions and calls to action) include and transcend reports and commentaries.

And that takes me to a suspicion that, I might be misreading/mishearing diSessa’s account of computational literacy. My desire for clearer definition tells me that I’m probably forcing his presentations of the construct into a descriptive report, and it strikes me that the notion (and diSessa’s presentations) might be better considered as commentary, perhaps even manifesto.

### Am I Hearing Correctly?

So, I’ll run with that thought—that is, that the construct of “computational literacy” is in essence a call to action. Accepting that computational literacy will be “close to universality” (diSessa, 2018), what should we be doing? DiSessa has (2018) has suggested a starting place: Get to know computational media as deeply as you can ... the most epistemologically rich computer systems you can find. By “epistemologically rich,” I mean having legitimate—



but likely as-yet unrealized—consequences for the mathematics we can experience and might teach. (p. 39)

Clearly, that's sensible advice, and it feels like it's driven by the sort of optimism that we educators must occupy. In particular, I very much appreciate that the gaze here is on an as-yet unrealized future, not an established past.

But, at the risk of raining on the parade, the call was published five years ago ... and, given the ever-accelerating pace of technological development, I'm mindful that the world of formal education has almost certainly fallen even further behind. Curriculum, especially, doesn't seem inclined to budge—except, perhaps ironically, around the topic of coding, which is doubtless at the core of the conflation of computational thinking and computational literacy.

Much as I don't want to sink into pessimism, I do feel a need to call for tempered optimism. This morning—on 2023 October 16, almost six months after the meeting at Brock—one of the top stories in my newsfeed was titled, "AI could spur an economic boom. Humans are in the way" (Omeokwe, 2023). Even after taking into account a likely lean toward the sensational here, I'd be lying to say I'm not dismayed at the prospects for computational literacy in light of where things seem to be going. Might the window of opportunity for this great idea have closed?

It should be clear by now, given my ongoing struggles to get a handle on computational literacy, that I'm not a quick study. That said, I want to mention that I did begin to feel some clarity as I read diSessa's (2018) focused commentary on Jeanette Wing's take (or *mis*-take?) on the topic. I'll abbreviate my interpretation by offering that Wing's account feels a whole lot like the authoritative-but-ultimately-incoherent ChatGPT explanation pasted above. Her version of computational literacy is a lasagna of HOTS (higher-order thinking skills), mental discipline, equity, liberation, competitive advantage, vocationalism, and inclusivity. DiSessa has offered a decidedly clearer vision.

Pushing aside my anxiety that the window of opportunity on that vision may have been missed, on the matter of what we as educators should be doing around computational literacy, I'm landing on the thought that our most important work is to avoid being "Winged." And, for that, I thank diSessa. As much as I struggle with the construct of computational literacy, I recognize it as a necessary part of formal education's obligation to help its society maintain fitness with the evolving world.

### References

- Davis, B. (2019). Methodological pluralism and graduate research in education. In V. Bohac-Clarke (Ed.), *Integral theory and transdisciplinary action research in education* (pp. 1–18). IGI-Global.
- diSessa, A. A. (2018). Computational literacy and "The Big Picture" concerning computers in mathematics education. *Mathematical Thinking and Learning*, 20(1), 3–31. DOI: 10.1080/10986065.2018.1403544
- Omeokwe, A. (2023 October 16). AI could spur an economic boom. Humans are in the way. *The Wall Street Journal*. Available at: <https://www.wsj.com/tech/ai/ai-could-spur-an-economic-boom-humans-are-in-the-way-3f13182c#>



## **Coded Bias: Decoding Racism in Artificial Intelligence Technologies**

Gideon Christian

Faculty of Law, University of Calgary

### **Introduction**

Artificial intelligence (AI) can simply be defined as the science and engineering of making computers perform tasks that, if performed by humans, would require human or cognitive intelligence (Scherer, 2016). AI is a broad field of scientific endeavor. This paper will focus on two aspects of AI technology where Black people and people of color should be really concerned about the use of this technology – that is in the area of AI recidivism risk assessment and AI facial recognition technology.

We are witnessing the increasing use of AI risk assessment in the criminal justice system, and AI facial recognition technology in public sectors – from immigration enforcement to criminal law enforcement and beyond. These technologies have been touted as major innovations in the field of artificial intelligence. However, their use comes with increasing concern about their potential to perpetuate race-based bias and discrimination against Blacks and people of color. This phenomenon has been variously referred to by Michele Alexander (2010) as “the new Jim Crow” and by Ruha Benjamin (2019) as “the new Jim Code.”

In criminal justice risk assessment, AI tools have been shown to assess Black defendants at a higher rate of recidivism, doing so at “twice the rate as White defendants” (Angwin et al., 2016, para. 15). Similarly, AI facial recognition technology has been shown to have a higher tendency to misidentify Black people and people of color (Najibi, 2020), leading to wrongful arrests and detentions in some cases. Thus, many AI tools appear to exhibit inherent bias against Black people and people of color (Johnson, 2022). This prompts the question: How and why do these tools demonstrate inherent bias against specific racial groups?

### **Algorithm and Bias**

There are many ways biased outcomes could result from AI tools. First, when an AI tool is deliberately designed to generate biased outcomes. This may be rare but is a clear case of explicit bias resulting from bad faith software engineering. The second situation is when biased data is used to train the computer algorithm to make predictions. This would result in a biased outcome, an extension of what is referred to in the computer lexicon as “garbage in, garbage out”. The third situation arises when an AI tool trained on data that is responsive to unique factors in a particular environment is deployed for use in a different environment. The second and third cases above would generally result in implicit bias.

### **AI in Criminal Justice Recidivism Risk Assessment**

Recidivism risk assessment is the process of determining the likelihood that an individual in the criminal justice system – an accused, convicted, or incarcerated person – will reoffend (Christian, 2020). Recidivism risk assessment tools that rely on AI technology are now increasingly being used in the criminal justice systems around the world. Research studies on the use of AI in criminal justice risk assessment have revealed the tendency of AI technologies to automate inequality and perpetuate bias, discrimination, and stereotypes against minority populations characterized by race (Angwin et al., 2016; Chouldechova, 2016).



In Canada, these minority groups are disproportionately represented in the criminal justice system. The unfair representation of Blacks and Indigenous people in the Canadian criminal justice system is well-documented (see *R. v. Jackson*, 2018). Although Blacks represent 4% of the adult population in Canada, they currently constitute 9% of the total federal in-custody population (Department of Justice Canada, Research and Statistics Division, 2022). Similarly, the over-representation of Indigenous persons in correctional facilities in Canada increased by 14% in 2020/2021 (Robinson et al., 2023). Although AI risk assessment tools are not currently being used in Canada, their use in the US criminal justice system is very pronounced. Hence, it is only a matter of time before Canada follows the trend. Thus, there is concern that the use of the tools in Canada may eventually automate the already existing racial bias and inequity evident in the Canadian criminal justice system.

In 2014, then US Attorney General Eric Holder warned of the danger in risk score injecting bias into the court system (Holder, 2014). He noted that while these risk assessment tools might have been designed with the best intention, they inadvertently undermine efforts to ensure justice by exacerbating biases that already exist in the criminal justice system (Holder, 2014).

The fear expressed by Eric Holder was confirmed by a research study undertaken by the US non-profit newsroom ProPublica, which revealed the tendency of algorithmic risk assessment tools to perpetuate existing racial bias and stereotypes in society (Angwin et al., 2016). The study focused on recidivism risk predictions by COMPAS – the most widely used AI risk assessment tool in the US criminal justice system. In addition to raising some doubts about the accuracy of predictions made by this risk assessment tool, the study raised even more serious concerns regarding the racial bias evident in COMPAS risk predictions. The study revealed that COMPAS not only falsely assesses Black defendants as future criminals but again does so twice as often as White defendants (Angwin et al., 2016). In fact, White defendants were more often misclassified as “low risk” than Black defendants. In some cases, Black defendants labeled by COMPAS as “high risk” did not reoffend, while White defendants labeled as “low risk” went on to reoffend. So, the study from ProPublica seems to establish that the use of COMPAS in risk assessment results in White defendants getting an unfair pass while Black defendants, on the other hand, get an unfair penalty. It is surprising then that, notwithstanding these findings, the COMPAS AI risk assessment software remains one of the most dominant AI-based risk assessment tools in the US criminal justice system.

### **Algorithmic Racism**

The tendency of results from AI risk assessment tools to be biased against people of colour has been referred to as algorithmic racism, which is defined as “systemic, race-based bias arising from the use of AI-powered tools in the analysis of data in decision making resulting in unfair outcomes to individuals from a particular segment of the society distinguished by race” (Christian, 2020, para. 3). Algorithmic racism may result from the use of historical data collected from the era of biased policing and carceral justice system to develop or train AI technology used in risk assessment.

The fact, though, is that AI risk assessment cannot be better than the data used to train or develop the tools used in the assessment. The problem with using data collected from these discriminatory carceral and law enforcement practices is that individuals from minority communities are disproportionately represented in the data, thus implying that those communities are more prone to crime or more likely to commit crime. An AI tool trained on such data will

inevitably regurgitate the bias and stereotypes implicit in the data. Such bias would even become more blurred by a false belief in technology as being race-neutral and color-blind.

### **Algorithmic Bias – Square Peg in a Round Hole**

Algorithmic bias would also arise when an AI tool developed and trained on data from one predominant racial group is sought to be used or deployed for use in another racial group not adequately represented in the training data. This problem has been evident in facial recognition software, which is discussed below. To illustrate this situation in the context of criminal justice risk assessment, consider the Supreme Court of Canada case of *Ewert v. Canada* (2018). In *Ewert*, an Indigenous man was serving consecutive life sentences. While he was serving his sentence, Correctional Services Canada (CSC) reviewed and upgraded his security classification after assessing his risk using an actuarial risk assessment tool.

The tools used by CSC were developed and tested on a predominantly non-Indigenous population. Applying such tools to assess the risk posed by an Indigenous person – a person from a racial group not adequately represented in the training data used to develop the tool – would reasonably raise some serious concerns as to the accuracy or validity of any assessment made by such tools. The Supreme Court in *Ewert* (2018) noted that such cross-cultural application of risk assessment tools would result in “cultural bias” (para. 71). Hence, deploying AI tools for use in an environment different from those for which the tools were originally and purposefully designed will result in biased and inaccurate predictions of risk.

AI risk assessment tools are not one-size-fits-all. These tools must be confined to the environment or groups for which they were originally designed. Efforts should also be made to ensure that even in the environment for which the tools are designed or created, the population in that environment is adequately represented in the training data used to develop the technology. Thus, representation is tied to the accuracy or lack of accuracy in the predictions made by AI tools.

### **Facial Recognition Technology – The New Jim Crow**

Jim Crow laws were a series of state and local laws in the United States that enforced racial segregation and discrimination against Black people and people of color from the late 19th century until the mid-20th century. These laws essentially denied recognition to Black people in certain spaces, as evidenced by their institutionalization of segregation, among other discriminatory practices. The denial of recognition to these groups is also evident today in facial recognition technology, which struggles to accurately recognize Black faces, as evidenced by its high error rate in recognizing Black faces, especially Black women.

Realistically, Jim Crow laws differ from AI facial recognition technology in the sense that the former was a deliberate state policy, and the later is not. However, both phenomena are similar in that they both involve systemic discrimination against Black individuals, they both result in the perpetuation of discrimination which disproportionately affects Black individuals, and the end result is that Black individuals bear the brunt of the negative consequences in both cases.

The adverse impacts resulting from the use of facial recognition technology in law enforcement are evident from various incidents in the United States of wrongful arrest and detention of Black people as a result of false identification by AI facial recognition technology (Johnson, 2022). An example is the case of Nijeer Parks, a Black man from New Jersey who in 2018 was wrongfully arrested and charged with serious crimes after facial recognition technology incorrectly matched his photo to that of the perpetrator of the crime (General &





Sarlin, 2021). He spent 10 days in jail, and it took almost a year before the charges against him were dropped after he found a transaction receipt which placed him 30 miles away from the crime scene, thus providing a perfect alibi. Nijeer's case represents one among many others in which innocent Black people have been wrongly arrested and, in some cases, falsely charged and locked up for crimes they never committed, all as a result of false identification by AI facial recognition technology.

### **AI Facial Recognition Technology in Canada**

For an unknown period of time, AI facial recognition technology was deployed and used by various law enforcement agencies in Canada, including the RCMP and Toronto Police (see Boutilier, 2021). The clandestine use of AI facial recognition technology by these law enforcement agencies sparked outrage when it became public that they were using the controversial Clearview AI software. This resulted in a joint investigation by the Office of the Privacy Commissioner of Canada, Commission d'accès à l'information du Québec, Information and Privacy Commissioner for British Columbia, and Information Privacy Commissioner of Alberta (2021) and even a Parliamentary committee hearing (Standing Committee on Access to Information, Privacy and Ethics, 2022). It is important to note, though, that the outcry against the use of AI facial recognition technology by these public agencies in Canada was not based on the racist nature of the technology – that is, its high propensity to misidentify people of color. Rather, the outcry related to the unethical and indeed unlawful methods adopted by the developer in scraping the internet for people's photos without their consent and using such photos to build their database. Hence, a case of privacy breach/concern.

Aside from the privacy issue evident from the Clearview software, which was concerning to many White people, Black people and people of color indeed have even more concerns – the high error rate in the identification of individuals from these racial groups by the technology. This is what critical race theory refers to as interest convergence – a situation where the interest of the minority group is advanced because it aligns or converges with that of the dominant group.

Unlike the US, in Canada, we do not have any highly publicized cases of Black people being wrongfully arrested because of misidentification by AI facial recognition technology. While we cannot categorically state that such incidents never existed in Canada, it is important to note that police use of this tool in Canada was clandestine. Hence, even where there might have been actual cases of wrongful arrests arising from misidentification by the police use of the tool in Canada, the use of such information in effecting the arrest might have been concealed by the same police department secretly using the technology.

However, the use of facial recognition technology in the Canadian immigration system began to gather attention with the publication of the Federal Court of Canada decision in the case of *Barre v. Canada (Citizenship and Immigration)* (2022). This case involved two Black Somali women who made a successful refugee claim in Canada. The Minister of Public Safety later sought to revoke their refugee status on the ground that their faces matched those of other women who entered Canada as Kenyan citizens. The women challenged the revocation of their refugee status at the Federal Court. While the women asserted that the government used the discredited Clearview software in matching their faces to those of other women, the Minister refused to disclose whether it used the technology, asserting that the information was protected by law. The Federal Court not only accepted the women's assertion that the facial recognition technology was used by the government, but the court also held that the disclosure of information relating to the government's use of the technology here was not protected by law (the



Privacy Act). Referring to facial recognition software as an “unreliable pseudoscience,” the court also expressed a great deal of concern about the use of the technology by the government on the women, considering “that darker-skinned females are the most misclassified group with error rates of up to 34.7%, as compared to the error rate for lighter-skinned males at 0.8%” (*Barre v. Canada*, 2022, para. 25; see also Lohr, 2018).

Unfortunately, while *Barre* was the first, it was not the last reported case of Canadian government agencies seeking to use evidence from facial recognition technologies to revoke the refugee status of successful refugee claimants of Black race. In fact, to date, all reported Federal Court cases in which the Canadian government has sought to use evidence from facial recognition technology to revoke successful refugee claims have involved individuals of Black race and predominantly Black women (in addition to *Barre vs. Canada*, 2022, see also *AB v. Canada*, 2023; *Abdulle v. Canada*, 2023; *Ali v. Canada*, 2023). The same racial/gender group where the technology has its highest error rate.

What is also concerning about the use of this technology by the Canadian government agencies is the lack of transparency in its use by the government. In all the Federal Court cases to date, the government agencies have refused to even admit their use of the technology. Rather, they attempt to rely on legal provisions (though unsuccessfully) to avoid the disclosure of such information. AI facial recognition technology is a Blackbox, responsible government use of the technology in context affecting the rights of citizens (irrespective of race) should be characterised by openness and transparency.

While ethical guidelines for the use of AI facial recognition technology are still evolving, the following common-sense principle should guide every public sector use of the technology in Canada: When the use of the technology affects an individual's rights, the government agency must inform the impacted individual and provide them with an opportunity to challenge the decision made by, or using the technology. Hopefully, the government agencies will incorporate this common-sense principle into their use of the technology to achieve the measure of transparency and openness that is characteristic of the system of government in Canada.

### **Conclusion**

Algorithmic bias impeded in AI technologies plays a significant role in generating unfair outcomes along racial lines. Whether arising from deliberately designed biased algorithms, biased training data, or the misapplication of tools to different racial groups not adequately represented by the training data, the result is the same: perpetuation of discrimination and inequality. To achieve accurate and just results, AI tools must be developed and trained with adequate representation from the communities they will affect. Ignoring this crucial aspect leads to systemic discrimination, as evidenced in the cases of AI risk assessment tools and facial recognition technology.

To ensure responsible and ethical use of AI facial recognition technology, transparency and openness must be embraced by Canadian government agencies. Public sector use of the technology should prioritize informing individuals impacted by its use and allowing them an opportunity to challenge any decisions made based on the technology's outcomes. This openness and transparency will enable the decoding of racial bias imbedded in AI technologies.

### **References**

AB v. Canada (Citizenship and Immigration), 2023 FC 29 (CanLII). <https://canlii.ca/t/jts52>



- Abdulle v. Canada (Citizenship and Immigration), 2023 FC 162 (CanLII).  
<https://canlii.ca/t/jvdcp>
- Alexander, M. (2010). *The new Jim Crow: Mass incarceration in the age of colorblindness*. The New Press.
- Ali v. Canada (Citizenship and Immigration), 2023 FC 671 (CanLII). <https://canlii.ca/t/jx64r>
- Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016, May 23). Machine bias. *ProPublica*.  
<https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing>
- Barre v. Canada (Citizenship and Immigration), 2022 FC 1078 (CanLII). <https://canlii.ca/t/jr6r8>
- Benjamin, R. (2019). *Race after technology: Abolitionist tools for the New Jim Code*. Polity.
- Boutilier, A. (2021, June 10). RCMP broke privacy laws in using controversial Clearview AI facial recognition tools, watchdog says. *Toronto Star*.  
[www.thestar.com/politics/federal/2021/06/10/rcmp-broke-privacy-laws-in-using-controversial-clearview-ai-facial-recognition-tools-watchdog-says.html](http://www.thestar.com/politics/federal/2021/06/10/rcmp-broke-privacy-laws-in-using-controversial-clearview-ai-facial-recognition-tools-watchdog-says.html)
- Chouldechova, A. (2016, October 24). Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *arXiv*. <https://doi.org/10.48550/arXiv.1610.07524>
- Christian, G. (2020, October 26). Artificial intelligence, algorithmic racism and the Canadian criminal justice system. *Slaw*. <https://www.slaw.ca/2020/10/26/artificial-intelligence-algorithmic-racism-and-the-canadian-criminal-justice-system/>
- Department of Justice Canada, Research and Statistics Division. (2022, December). Overrepresentation of Black people in the Canadian criminal justice system. *JustFacts*.  
[https://www.justice.gc.ca/eng/rp-pr/jr/obpccjs-spnsjpc/index.html#\\_ftnref26](https://www.justice.gc.ca/eng/rp-pr/jr/obpccjs-spnsjpc/index.html#_ftnref26)
- Ewert v. Canada, 2018 SCC 30, [2018] 2 S.C.R. 165. <https://scc-csc.lexum.com/scc-csc/scc-csc/en/item/17133/index.do>
- General, J., & Sarlin, J. (2021, April 29). *A false facial recognition match sent this innocent Black man to jail*. CNN. <https://www.cnn.com/2021/04/29/tech/nijeer-parks-facial-recognition-police-arrest/index.html>
- Holder, E. H., Jr. (2014, August 1). *Attorney General Eric Holder speaks at the National Association of Criminal Defense Lawyers 57th annual meeting and 9th State Criminal Justice Network conference* [Speech]. <https://www.justice.gov/opa/speech/attorney-general-eric-holder-speaks-national-association-criminal-defense-lawyers-57th>
- Johnson, K. (2022, March 7). *How wrongful arrests based on AI derailed 3 men's lives*. *Wired*.  
<https://www.wired.com/story/wrongful-arrests-ai-derailed-3-mens-lives/>
- Lohr, S. (2018, February 9). Facial recognition is accurate, if you're a White guy. *The New York Times*. [www.nytimes.com/2018/02/09/technology/facial-recognition-race-artificial-intelligence.html](http://www.nytimes.com/2018/02/09/technology/facial-recognition-race-artificial-intelligence.html)
- Najibi, A. (2020, October 24). Racial discrimination in face recognition technology. *SITN*.  
<https://sitn.hms.harvard.edu/flash/2020/racial-discrimination-in-face-recognition-technology/>
- Office of the Privacy Commissioner of Canada, Commission d'accès à l'information du Québec, Information and Privacy Commissioner for British Columbia, and Information Privacy Commissioner of Alberta. (2021, February 2). *PIPEDA Findings #2021-001, Joint investigation of Clearview AI, Inc.* <https://www.priv.gc.ca/en/opc-actions-and-decisions/investigations/investigations-into-businesses/2021/pipeda-2021-001/>
- R. v. Jackson, 2018 ONSC 2527 (CanLII). <https://canlii.ca/t/hrm8w>



Christian, C. (2023, April 27–29). Coded Bias: Decoding Racism in Artificial Intelligence Technologies. In Online Proceedings of the *Coding, Computational Modelling, and Equity in Mathematics Education Symposium*, St. Catharines (Canada), April 2023.

- Robinson, P., Small, T., Chen, A., & Irving, M. (2023, July 12). Over-representation of Indigenous persons in adult provincial custody, 2019/2020 and 2020/2021. *Juristat*. <https://www150.statcan.gc.ca/n1/en/catalogue/85-002-X202300100004>
- Scherer, M. U. (2016). Regulating artificial intelligence systems: Risks, challenges, competencies, and strategies. *Harvard Journal of Law and Technology*, 29(2), 354–400. <https://doi.org/10.2139/ssrn.2609777>
- Standing Committee on Access to Information, Privacy and Ethics (ETHI). (2022, October). *Use and impact of facial recognition technology: Report of the Standing Committee on Access to Information, Privacy and Ethics*. <https://www.ourcommons.ca/DocumentViewer/en/44-1/ETHI/report-6>



## Anti-Bias and Pro-Transformation: How to Merge Critique and Transformative Visions for Artificial Intelligence

Ron Eglash  
School of Information, University of Michigan

As our keynote speaker Gideon Christian points out, the dangers of bias in AI and other data-intensive information sciences have been well documented (Angwin et al., 2022). They include risk prediction equations used by criminal justice officials to inform their decisions about bail, sentencing and early release; bank loans, medical decisions, and many other aspects of our lives. But an exclusive focus on “bias” is not enough, we need to be both anti-bias and, simultaneously, create transformative change.

What is the difference? If we focus exclusively on eliminating bias, we imply that if only the bias would vanish, we would have a just and equitable system. But that is not at all the case. For example, our current banking algorithms have resulted in higher loan rates for Black home buyers, because of bias in the ways they calculate risk. But that bias does not address the problem that homes and loans are extremely expensive to begin with. For the working class, even in the absence of bias, the dangers of defaulting on loans are significant. They have been a widespread destructive force in working class communities, no matter what color. A system designed to make the rich even richer, at the expense of the working poor, does not need bias to enact forms of oppression. An exclusive focus on eliminating bias can thus become a distraction from the more important project of transformation. Additional examples appear in table 1.

*Table 1: The contrast of anti-bias computing and its critique from a transformative perspective.*

anti-bias computing	transformative critique
“We should eliminate racial bias in face recognition systems”	<b>Yes, but</b> that just helps “officials to become more adept at criminalizing Black people” (quote from Ruha Benjamin). What we need is restorative justice.
“We should eliminate bias in prison sentencing”	<b>Yes, but</b> the US has 20% of all prisoners on earth. Eliminating bias would barely change it. What needs to change is the prison-industrial complex.
“Advertisements targeted to poor people lack opportunities for investment”	<b>Yes, but</b> if you do not have money to invest, that is no help to your finances. What needs to change is the economic system of value extraction.

Within STEM education, the same contrast can be applied. It's great to have teachers who are interested in ensuring that instruction is not biased. But for children already challenged by social inequity, merely eliminating bias is not sufficient. And in terms of course content, introducing subjects like algorithmic bias, or the bias towards military applications, or similar



kinds of critiques in computer science classrooms might help alert students to potential problems, but focusing on negative critiques may give the impression that if you do go into computing, you should be expected to set aside social justice issues. Focusing on negative critiques does not engage student’s agency and imagination in developing new technologies as a means of social transformation.

In contrast, our research group has been developing three domains in which a more positive approach to computing in the service of social transformation can be achieved, including in STEM education. These are ethnocomputing in Indigenous knowledge; the use of heritage algorithms for low-income community development; and the application of AI in the solidarity economy.

A good “example to think with” as Levi-Strauss would say is African fractals (Eglash 1999). Looking at aerial photos of African villages (figure 1), we can see circles of circles, rectangles of rectangles, rings of rings. A year of fieldwork in west and central Africa showed that the builders were not simply creating unconscious patterns: these were “heritage algorithms” that enacted deeply held beliefs and practices about the spiritual, ecological and social dimensions of the worlds as self-generating. Spiritual concepts such as the ancestors of ancestors, ecological concepts such as the feedback loop between nature and humans, and social concepts such as the iterations of reciprocity between individual and community, are all embedded in these forms.

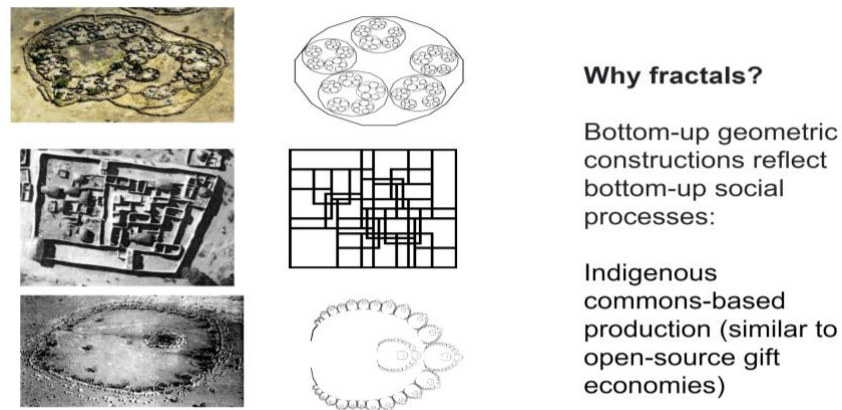


Figure 1. Fractal simulations of traditional African architecture.

The computational forms of recursion, central to disciplines such as complexity theory, network evolution studies, models of self-organization in nature, generative AI and similar branches of knowledge are thus equally of central importance in these African traditions, where they represent the circle of life, the spiritual self-emergence of fecundity and the obligations of evolving kinship networks across time, space and species. They are powerful sources of insight into how we might create contemporary forms of circular economies, justice-based societies and ecological balance, if we can learn to “translate” these into technological domains.

Our first step in doing that translation was in STEM education. Readers can go to the Culturally Situated Design Tools website ([csdt.org](http://csdt.org)) and explore African fractals, iterative design in cornrow braiding, Native American biocomplexity, Artificial Intelligence for community development and many other frameworks for transforming our classrooms into laboratories in which students explore ethnocomputing and ethnomathematics (figure 2). These practices can show statistically significant improvement in underrepresented student interest and performance (Eglash et al., 2020; 2021; Lachney et al., 2021).

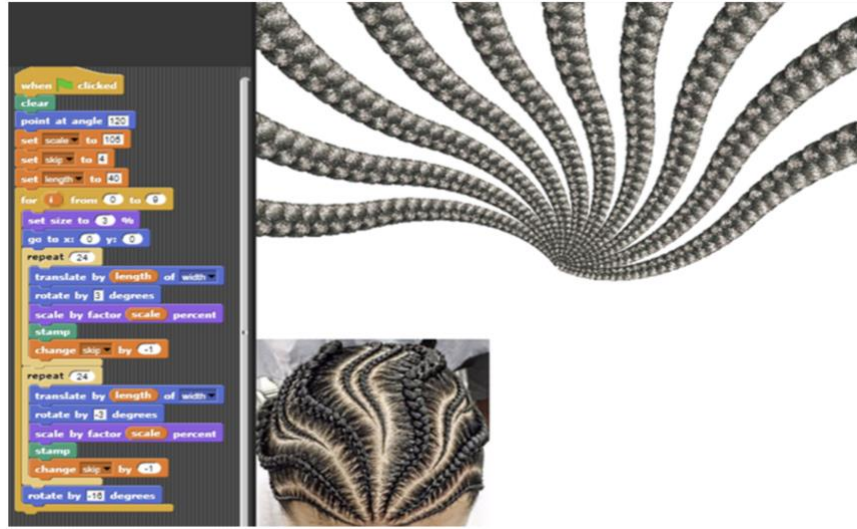


Figure 2. Programming interface for CSDT based on cornrow hairstyles.

The next step in our exploration was to look at the circular economy offered by those African traditions, and consider how similar kinds of unalienated value flow (ie “generative justice”) might be implemented in relation to the adult economy. We brought some professional braiders into conversation with the team, and developed new extensions based on problems they located, such as pH damage to hair and the need to bring in new customers. Out of that came our first investigations into applications for economic value (figure 3).

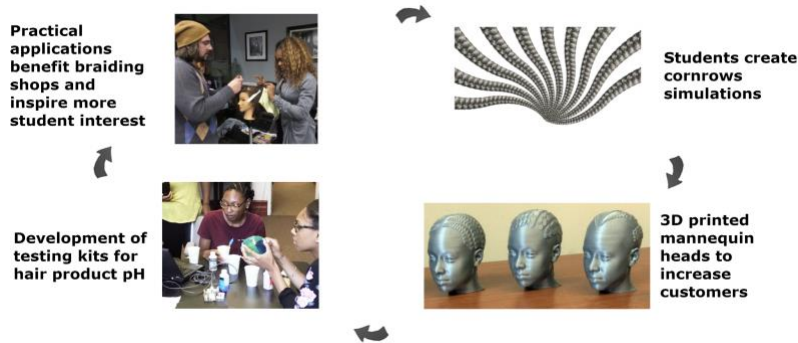


Figure 3. A circular economy connecting STEM and worker-owned braiding shops.

More recently, we obtained an NSF grant, “Race, Gender and Class Equity in the Future of Work: Automation for the Artisanal Economy”. Here we are applying AI, digital fabrication, and a broad array of other technologies to examine how more extensive networks of low-income community entrepreneurship, urban farming, civic organizations and others might be developed as a community-based economy (figure 4). For example, e-delivery by DoorDash, UberEats or other services charge significant amounts of overhead: much of the profit does not go to the drivers or product providers. So we are working with Detroit’s urban farmers to develop an independent system that they can own, control, and co-design with us, so that AI and related algorithmic services can optimize driving routes for their good, not merely the priorities of some distant corporation. AI can similarly be used to help authenticate artisanal work from factory fakes; help consumers find local options to replace industrial farmed produce or overseas products, and many other strategies for keeping value at the grassroots where the actual work is

being done, both by humans and our non-human allies in nature. Other work includes automating fabrication for clothing, furniture, and so on, so that local artisans can put more time into their own creative activities, while retaining ownership. Those interested in K-12 education based on such efforts can use the CSDT at [https://csdt.org/workbooks/start\\_aikr\\_compare](https://csdt.org/workbooks/start_aikr_compare). For the adult economy itself, our publications are available at <https://generativejustice.org/publications>.

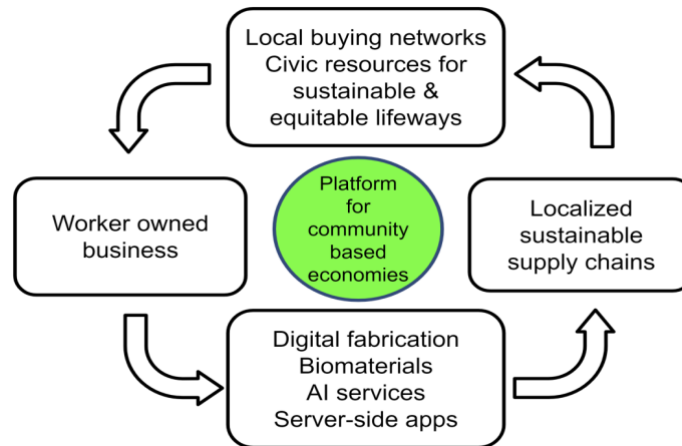


Figure 4. Computation for a community-based economy.

In conclusion: the elimination of bias is an important part of AI regulation. But alone it is insufficient. We need education, research and policy development that can utilize emerging technologies to offer a transformative move to social, economic and civic systems for generative justice.

### Acknowledgement

This material is based upon work supported by the National Science Foundation (NSF) Future of Work at the Human-Technology Frontier under Grant No. 2128756. All opinions stated or implied in this document are those of the authors and not their respective institutions or the National Science Foundation.

### References

- Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2022). Machine Bias. In *Ethics of Data and Analytics* (pp. 254-264). Auerbach Publications.
- Eglash, R. (1999). *African Fractals: Modern Computing and Indigenous Design*. Routledge.
- Eglash, R., Bennett, A., Babbitt, W., Lachney, M., Reinhardt, M., & Hammond-Sowah, D. (2020). Decolonizing posthumanism: Indigenous material agency in generative STEM. *British Journal of Educational Technology*, 51(4), 1334-1353.
- Eglash, R., Bennett, A., Cooke, L., Babbitt, W., & Lachney, M. (2021). Counter-hegemonic computing: Toward computer science education for value generation and emancipation. *ACM Transactions on Computing Education (TOCE)*, 21(4), 1-30.
- Lachney, M., Babbitt, W., Bennett, A., & Eglash, R. (2021). Generative computing: African-American cosmetology as a link between computing education and community wealth. *Interactive Learning Environments*, 29(7), 1115-1135



## **How Do Computational Thinking and Mathematical Thinking Interact (In Terms of Knowledge, Ways of Thinking and Competencies)?**

Eirini Geraniou<sup>1</sup>, Paul Drijvers<sup>2</sup>, & Elise Lockwood<sup>3</sup>,

<sup>1</sup>IOE, UCL's Faculty of Society and Education, University College London, UK, e.geraniou@ucl.ac.uk; <sup>2</sup> Freudenthal Institute, Utrecht University, the Netherlands, p.drijvers@uu.nl; <sup>3</sup> Oregon State University, USA, elise.lockwood@oregonstate.edu

### **Introduction**

Over the past decade, there has been an upsurge of interest in the research on computational thinking (CT). Many countries are in the process of introducing computational thinking into their school curricula, either as a new dedicated subject, a cross-curricular theme, or integrated within an existing subject, such as computing (e.g., Kafai & Proctor, 2022) or mathematics (e.g., Bocconi et al., 2018; Royal Society, 2018; Nordby, Bjerke & Mifsud, 2022). The relationship between computational thinking and mathematical thinking has been of particular interest (e.g., Perez, 2018, Kallia et al., 2021, Geraniou & Hodgen, 2022a; 2022b) and there is variation in how computational thinking is perceived by different stakeholders in the mathematics education discipline, including researchers, teacher practitioners, and teacher educators.

In this interactive research panel, the panellists discussed characterizations of computational thinking and how it relates to mathematical thinking in terms of knowledge, ways of thinking, and competency. There were two sessions. The first session started off with an introductory task for the audience, seeking participants' past experiences with CT in mathematics, their views on CT's importance for different stakeholders from the mathematics education broader community and its relevance to today's society, all of which were recorded using a padlet. This activity was intended to set the scene for participants and supported them in reflecting on what followed in this panel session. Paul addressed the importance of computational thinking in mathematics education in the 21st century and its potential links to mathematical thinking from an historical, research-based perspective. Elise and Eirini reacted by sharing their own positionalities and personal interests, what they found interesting, and what future work in this field should entail. This first session ended with the panellists responding to some ideas shared in the padlet. The second session focused on the presentation of two engaging, yet contrasting, vignettes regarding the impact of CT in mathematics education, envisioning what is to happen in the year 2040 in a mathematics classroom and in the mathematics education scene in general from a societal, educational and political perspective. The participants were asked to use another padlet to respond to vignette-specific reflective questions, but also some general questions: 1. Does the scenario in this vignette excite you? Concern you; 2. Are constructs like "computational thinking" or "mathematical thinking" evident in this vignette? If so, what are the similarities or differences between them? The panellists were then invited to react to the points raised in the padlet. The panel session ended with each panellist sharing their one-line take-home message that captured their key reaction.

In the next pages, we present our gathered reflections from the various parts of this panel session, highlighting questions and issues related to computational thinking that, in our view, are of great interest to the mathematics education research community.



## How Do Computational Thinking and Mathematical Thinking Interact?

Why are we interested in Computational Thinking? As Paul argued, our private and professional lives are full of interactions with digital technology that are based on computations. As (future) citizens and professionals, we need to be able to deal with this and ensure we have a good insight into the computational processes involved. But one could wonder whether CT is really a “new” concept. Papert mentioned CT in his *Mindstorms* book and argued that “...visions of how to integrate computational thinking into everyday life was insufficiently developed” (Papert, 1980, p.182). Most of us have seen the definitions of CT offered by various authors, For example, Wing (2010) defined CT as “the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent” (Cuny, Snyder & Wing, 2010). Lodi (2020) characterised CT as the “*disciplinary way of thinking*” of Informatics and argued that computational problem solving involves the formulation and the solution of the problem expressed in a way that allows an “external” processing agent (a human or a machine) to carry it out.

Some research also indicated the various processes involved with CT. Angeli et al. (2016) argued that CT involves five processes: Abstraction, Generalization, Decomposition, Algorithms, and Debugging. Some of these processes have been mentioned by Grover and Pea (2013) and Selby and Woollard (2015), who added a few more: Problem-solving, Abstraction, Generalization, Decomposition, Algorithmic thinking and Modularization. But how do computational thinking and mathematical thinking relate? Kallia et al. (2021) claimed that CT in mathematics education is seen as “a structured problem-solving approach in which one is able to solve and/or transfer the solution of a mathematical problem to other people or a machine by employing thinking processes that include abstraction, decomposition, pattern recognition, algorithmic thinking, modelling, logical and analytical thinking, generalisation and evaluation of solutions and strategies” (Kallia et al. 2021, pp.20-21). Considering such arguments, should we all agree that Mathematical Thinking (MT) is a subset of CT (Fig 1a)? Or do CT and MT have an intersection (Fig 1b)? Or is CT a subset of MT (Fig 1c)?

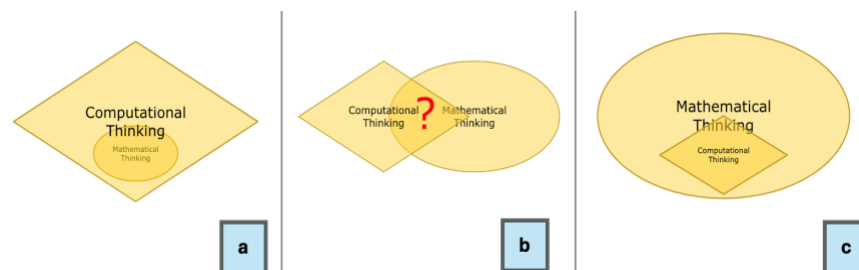


Figure 1. How do CT and MT relate?

So, which of these three reflects the best view? And what about the potential links between CT and programming and coding, or even artificial intelligence? As Paul concluded, the landscape of CT in education is still quite scattered, and even though CT and MT share common ground, it is still unclear how to cultivate that common ground.

Eirini reacted to Paul’s ideas by posing a question that has been in her mind for the past few years: “Why is CT *‘important’* in mathematics education and mathematics education research?” and in fact “why does it seem to be appearing in discussions within the mathematics education community and beyond?”. Her own research interests regarding computational





thinking focus on: (a) Investigating mathematics educators and mathematics teachers' views and perceptions on CT and its impact on mathematical learning; and on (b) identifying strategies for promoting CT and its integration in mathematics teaching and learning. In order to look at these 2 research foci, Eirini looked into what computational thinking is, considering some different definitions to those presented by Paul. It has been characterized as an essential competency for a digital society (Inprasitha, 2021) or the “new digital age competency” (e.g., Grover & Pea, 2013). It has been defined as “the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.” (Cuny, Snyder and Wing, 2010). It involves practices that are also required in mathematics, such as “decomposition, abstraction, algorithm design, debugging, iteration, and generalization” (Li et al., 2020, p.156). Perez’s (2018) review of CT in mathematics education highlights a range of practices, including elements, such as “developing algorithms and automations” as well as composite practices, such as “efficient and effective combinations of resources, testing and debugging” (p. 428). As one notices, there are some highlighted words, such as “competency” (in fact a “new digital age competency”), “thought processes”, “practices”, “a range of practices”. Reflecting on these definitions, Eirini drew attention to the idea of whether CT is indeed a competency and how it relates to MT. Paul looked at these two being separate, but also having an intersection and sharing common ground. In Sands et al.’s (2018) research, CT was synonymous to doing mathematics using digital tools. So, are CT and MT only linked when doing mathematics using digital technologies? Recent work by Geraniou and Hodgen (2023) indicated that some mathematics educators struggle to articulate what the relationship between CT and Mathematics is and, as a result, view CT as closely tied to computers and other digital tools. Carrying out this research work with mathematics educators led them to think about the relationship between mathematical knowledge, thinking, competency and where CT fits in. Having in mind that CT was perceived in the literature as a “competency” as mentioned earlier, Eirini reflected on definitions regarding mathematical competence to identify potential links. In fact, she wondered about whether there is a link between CT and mathematical digital competency, which is a construct developed recently by Geraniou and Jankvist (2019). They argued that when solving a mathematical problem using a digital tool, having mathematical competency and mathematical knowledge and digital competency and knowledge of the digital tool equips you to solve the problem. Mathematical Digital Competency was defined as the interplay of mathematical competency and digital competency and the MDC framework was developed (for more information, please read Geraniou & Jankvist, 2019). Going back to the work of Geraniou and Hodgen (2023) on mathematics educators’ views on CT, Eirini shared some reflections regarding a recent study involving interviews with mathematics educators seeking to find out more about their view on what CT is and how it relates to MT. In those interviews, mathematics educators were asked to bring a mathematical task involving the use of any digital tool that they are familiar with and the interview was based on their chosen tasks. After presenting one of these case studies, Eirini argued that it became clear to them that possessing computational thinking is a competency, in fact an MDC, and the MDC framework was adapted (Geraniou & Hodgen, 2023). Eirini therefore concluded with the argument that CT may indeed be a Mathematical Digital Competency.

Elise shared her own research path regarding mathematical and computational thinking, before reacting to Paul’s contribution. Her research focuses on investigating undergraduate students’ reasoning about combinatorics (counting) problems, which is a specific mathematical topic and most of her work did not consider computing at all. Indeed, Elise noted that she has

little programming or computational experience. However, her motivation to engage with computing is driven by her mathematics education research, as a central finding of her work in combinatorics emphasizes the value of helping students focus on sets of outcomes. She elaborated on this by presenting and discussing a task: “How many three-digit sequences can be made from the numbers 1-6 that have no repeated digits?” Figure 2 demonstrates an example of how a small bit of Python code offers an insightful representation to consider a solution to this counting task. The four loops in the code represent the multiplicative process in permutations, and the conditional statement ensure that elements are not repeated.

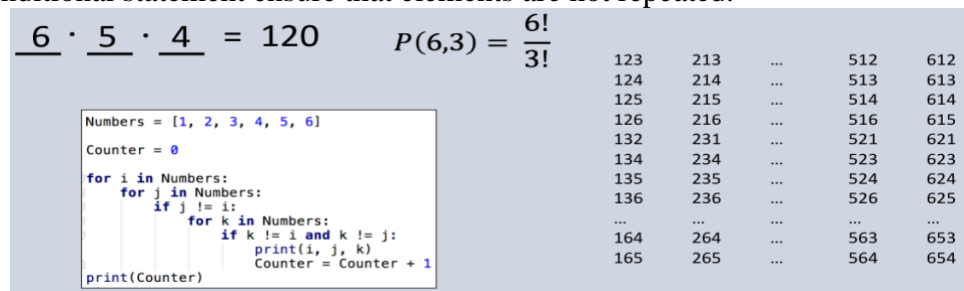


Figure 2. “How many three-digit sequences can be made from the numbers 1-6 that have no repeated digits?”

Elise used this task as an example of how CT and MT can intersect, and in her work she has argued that the two can be mutually beneficial; here, student reasoning about the computational representation of the code (an instance of CT) may support students’ understanding of combinatorial processes (an instance of MT). In this way, Elise finds herself aligning with Paul’s characterization in Figure 1b above, and her work seeks to focus on that intersection. She concluded by sharing her future research goals, which involve (a) exploring ways in which CT builds upon what we know about mathematics education research and (b) finding out if even modest engagement with CT and activity may support students’ MT and activity (and vice versa). She argued that she benefitted from engagement with computationally-minded researchers in other disciplines, and hopes for a future with meaningful interdisciplinary collaborations. She finally put forward a question for all to reflect upon: is there a future where computational and mathematical thinking reflexively develop and grow?

### The Importance and Value of Computational Thinking in Mathematics Education

The panellists wanted to inspire participants in considering the potential impact of CT in mathematics education and therefore carefully chose two engaging, yet contrasting, vignettes, envisioning what is to happen in the year 2040 in a mathematics classroom and in the mathematics education scene in general from a societal, educational and political perspective.

The first scenario was titled “The Adventure Travel List problem” and it stated “It’s year 2040 and the maths teacher sets the following homework task to their Year 9 class: “Harry needs to visit 5 different places in the world. He starts his adventure travel in London (UK) and then he needs to go to all the following 5 places: Kathmandu (Nepal), Nairobi (Kenya), Machu Picchu (Peru), Cairo (Egypt), Sydney (Australia). What is the shortest route that Harry can take to visit all 5 places and return to London?”, which is an adaptation of the well-known “[travelling salesman problem](#)”. The scenario presented the dialogue between two students, Mark and Sofia, and Mark was seeking help from Sofia to solve the problem. There was an element of surprise at the end as Sofia proved to be an Artificial Intelligent Tutor, programmed to support Mark, find a solution and reach an answer. Participants were asked to reflect on questions like: Do Mark and



Sofia have Mathematical Digital Competency? What about CT? Do they both have it? How would you feel about this task? Why would this be an activity worthwhile to teach in 2040? Would we have AI tutors supporting students to develop CT? Is there evidence of the interplay between CT and MDC in this vignette?

The second scenario involved a fictitious scene taking place in 2040, during which politicians recognised that having a computationally fluent workforce and citizenry was increasingly necessary for global competition, and were motivated to create governmental action plans to expedite the development of computing education. Twenty years later, in 2060, efforts to integrate computing into schools have broadly been implemented in many countries around the world. A graduation requirement from secondary school involves a computational data fluency certificate in any subject a student may choose, while at university level, most institutions for any major offer a comprehensive common first-year slate of courses that integrate computing with a student's discipline of choice. The education scene in 2060 looks very different. Computing and computational thinking has become the hub that connects all disciplines and academic journals favor interdisciplinary collaboration to represent excellence in scholarly educational research. At school, all students attend mathematics lessons for 20h/week, as it is widely recognized that in this era of computational thinking, mathematical skills are the only ones that matter for problem solving and for dialogue with intelligent computing tools. Classes are set up in traditional ways in old-fashioned classrooms, and they focus on ancient geometry, number theory, logic, and rhetorics, as to prepare students for the 22nd-century future. In some of these countries, this approach is only applied to the top-5% of high-achieving students, as they are identified in a problem-solving test for 4-year-old children. They are trained to become the engineers and scientific leaders for the future; being selected for this elitarian group is considered an honor. Participants were asked to reflect on questions like: From an instructional perspective, what might be gained or what would be lost in these two approaches? From a research perspective, what would be gained or lost in these two approaches?

The participants were also asked to respond to some general questions for each scenario: 1. Does the scenario in this vignette excite you? Concern you?; 2. Are constructs like “computational thinking” or “mathematical thinking” evident in this vignette? If so, what are the similarities or differences between them?

Both scenarios brought forward arguments about how realistic these cases are and participants raised some concerns about AI taking over the teacher's role for example (first scenario) or whether we would need to teach our students how to code if AI can do this for us (second scenario).

We, the panellists, see the value of both mathematical thinking and computational thinking, but are concerned about whether and how the two should be integrated. For example, one can go for the “pure and isolated” option and focus on MT, or for the embedded, real-life option of CT in applied situations, and perhaps competency (and potentially mathematical digital competency) should also be explored. We, as mathematics educators, should find ways, through our work and research, to influence any decisions taken by different stakeholders (e.g. government, policy-makers, schools and other institutions) regarding the integration of CT and MT. We know that digital technologies are part of our lives, AI is evolving, possibly even faster than anticipated, and any research we conduct will shape the mathematics education field in the coming decades. So, the key question to ask ourselves is: What should the mathematics education scene look like in the year 2050?



### Acknowledgments.

We are ever so grateful to Dr. Nathalie Sinclair (SFU, Canada), Canada Research Chair in Tangible Mathematics Learning, in her participation in early discussions for preparing for this panel. We are also very grateful to Ghislaine Guedet, for chairing the session and supporting us with the organisation of the panel discussions. Some of their ideas might be reflected in the panel structure as well as in the panellists' contributions.

### References

- Bocconi, S., Chiocciariello, A., & Earp, J. (2018). *The Nordic approach to introducing Computational Thinking and programming in compulsory education. Report prepared for the Nordic@BETT2018 Steering Group*: Nordic@BETT2018 Steering Group.
- Cuny, J., Snyder, L., & Wing, J.M. (2010). *Demystifying Computational Thinking for Non-Computer Scientists* (work in progress). Retrieved from <https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why>
- Geraniou, E., & Hodgen, J. (2022a). *An exploratory study on mathematics teacher educators' beliefs and understandings about computational thinking*. In J. Hodgen, E. Geraniou, G. Bolondi, & F. Ferretti (Eds.), *Proceedings of the Twelfth Congress of the European Society for Research in Mathematics Education (CERME12)*. Bolzano, Italy.
- Geraniou, E., & Hodgen, J. (2022b). *A case study of an expert in computational thinking in the context of mathematics education research*. In H-G. Weigand, A. Donevska-Todorova, E. Faggiano, P. Iannone, J. Medová, M. Tabach, & M. Turgut (Eds.), *MEDA3 Mathematics Education in the Digital Age 3: Proceedings of the 13th ERME Topic Conference (ETC3)*, 7-9 September 2022, Nitra, Slovakia (pp. 161-168). <https://hal.science/hal-03925304v1/>
- Kafai, Y. B., & Proctor, C. (2022). A Revaluation of Computational Thinking in K–12 Education: Moving Toward Computational Literacies. *Educational Researcher*, 51(2), 146–151. <https://doi.org/10.3102/0013189X211057904>
- Kallia, M., van Borkulo, S. P., Drijvers, P., Barendsen, E., & Tolboom, J. (2021). Characterising computational thinking in mathematics education: a literature-informed Delphi study. *Research in Mathematics Education*, 23(2), 159-187, <https://www.tandfonline.com/doi/full/10.1080/14794802.2020.1852104>
- Lodi, M. (2020). Informatical thinking. *Olympiads in Informatics*, 14, 113–132.
- Nordby, S. K., Bjerke, A. H., & Mifsud, L. (2022). Computational Thinking in the Primary Mathematics Classroom: a Systematic Review. *Digital Experiences in Mathematics Education*, 8, 27–49. <https://doi.org/10.1007/s40751-022-00102-5>
- Pérez, A. (2018). A framework for Computational Thinking Dispositions in Mathematics Education. *Journal for Research in Mathematics Education*, 49(4), 424–461. <https://doi.org/10.5951/jresmetheduc.49.4.0424>
- Royal Society. (2018). *The integration of data science in the primary and secondary curriculum*. Royal Society.
- Wing, J. M. (2010). Research notebook: Computational thinking – What and why. *The Link Magazine*, 20–23. Retrieved from <https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why>



## Mathematics Education Incorporating Coding: Practical Challenges and Opportunities

Celia Hoyles<sup>1</sup>, George Gadanidis<sup>2</sup>, Oh Nam Kwon<sup>3</sup>, Simon Modeste<sup>4</sup>, & Elena Prieto-Rodriguez<sup>5</sup>

<sup>1</sup>UCL Knowledge Lab, University College London, London, U.K.;

<sup>2</sup>Western University, Ontario, Canada, [ggadanid@uwo.ca](mailto:ggadanid@uwo.ca);

<sup>3</sup>Department of Mathematics Education, Seoul National University, Korea;

<sup>4</sup>Institut Montpelliérain Alexander Grothendieck, University of Montpellier & CNRS, France, [simon.modeste@umontpellier.fr](mailto:simon.modeste@umontpellier.fr);

<sup>5</sup>School of Education, University of Newcastle, Australia, [elena.prieto@newcastle.edu.au](mailto:elena.prieto@newcastle.edu.au);

### The Panel

The Panel was introduced by Dr Ana Isabel Sacristán Rock, Centre for Research and Advanced Studies (Cinvestav-IPN), Mexico.

This panel aimed to complement the research-focused panel held on the previous day, which focused on the question: *How do computational thinking and mathematical thinking interact in terms of knowledge, ways of thinking, and competencies?*

The following abstract provides an overview of the session along with a structure for the illustrative texts and videos presented by panelists, which are summarized at the end.

### Abstract

Coding or programming is ubiquitous across the world. But what coding means, how it is learned and developed and how it is exploited as a tool to explore topics in different subject areas varies enormously across jurisdictions: for example, coding can be introduced and developed as part of a school computing curriculum, as part of the school mathematics curriculum or more informally within out-of-school clubs. These different structural organizations inevitably serve to define what happens in practice in schools, shapes how students develop coding skills and learn key coding concepts, informs how teaching might be enhanced through coding, and ultimately how coding could be exploited outside of computing as a tool to think with and explore mathematics.

Each panelist set out to touch on these issues with a particular focus on the interactions between mathematics and coding *in practice* in their country. The discussion aimed to tease out the challenges, risks and opportunities of integrating coding into mathematics classrooms while addressing questions such as: Why should coding be incorporated in mathematics classrooms? What is the specificity of coding in mathematics and the links between coding and mathematics? What can coding bring to mathematics in terms of new content or new ways to support mathematics teaching and improve learning? Which mathematical topics are most aligned to the incorporation of coding and why? What are the links between coding and algorithmics, applied mathematics and reasoning?

The panel comprised a chair and four invited panelists from four different countries with very different curricular structures.

**Chair:** Prof Dame Celia Hoyles, United Kingdom

**Panelists:**

- Dr Oh Nam Kwon, Korea
- Dr Simon Modeste, France
- Dr Elena Prieto, Australia
- Dr George Gadanidis, Canada





**Reactor:** Prof Richard Noss, UCL Knowledge Lab, University College London, London, U.K.

The duration of the panel was 1.5 hours, following an introduction by Dr Ana Isabel Sacristán Rock. It comprised of:

- Celia Hoyles' introduction to the topic, the panel, and the panellists (5-10 min).
- Each panelist then briefly described their country's positioning of coding or programming in the school curriculum and any links with mathematics, illustrating their perspective on classroom implementation in their country with short videos, photographs or with student work (10 minutes each).
- The Chair and the panelists were then invited to provide short reactions to the presentations 5 mins each.
- Richard Noss then made his reaction to the panel session.
- Finally, questions were taken from the audience face-to-face or online.

Below is a brief summary of each of the presentations.

### **Mathematics Education Incorporating Coding: Practical Challenges and Opportunities in Korea.**

*Oh Nam Kwon, Seoul National University, Korea*

I explore the integration of coding into Korea's mathematics education, focusing on the AI Mathematics curriculum introduced in high school in 2020. This initiative, developed due to increasing societal demand for AI knowledge, emphasizes the relationship between AI and mathematics, data representation, categorization, prediction, and optimization, incorporating key mathematical concepts like vectors and matrices. Furthermore, I investigate into how coding, specifically block and text coding, is woven into this curriculum, aligning with Korea's national education strategy. This is exemplified through the classroom practices of Mr. Oh, a high school teacher who uses digital tools like Askmath AI Lab and Algeomath. Mr. Oh's experience highlights the importance of integrating mathematical concepts with AI applications, enhancing students' interest in both areas and preparing them for future technological challenges and STEM careers. The integration of AI and coding into mathematics education not only addresses practical challenges but also opens up significant opportunities for enriching students' learning experiences, blending traditional mathematics with modern technological applications. The innovative approach of integrating AI and coding into mathematics education in Korea, particularly through hands-on examples like those in Mr. Oh's classroom, represents a forward-thinking shift in educational practices. This paradigm shift not only makes mathematics more engaging and relevant for students but also bridges the gap between theoretical knowledge and real-world applications. This integration thus serves as a model for other educational systems worldwide, illustrating the benefits of adapting traditional curricula to include emerging technological trends.

### **An Example From an 'Ordinary' Situation in Grade 9 in France**

*Simon Modeste, University of Montpellier, France. [simon.modeste@umontpellier.fr](mailto:simon.modeste@umontpellier.fr)*

The aim of this presentation was to present an example from France of coding in mathematics in middle school (in grade 9), illustrated with a video.

#### **General Curricular Context**

To situate the context, Table 1 presents a synthesis of the current curricular situation in France, concerning mathematics and computer science. In primary school, computer science has been introduced as a subject since 2015 (note that in France, before grade 6, there is one teacher



for all the subjects). In middle school, computer science has been introduced inside two subjects: *technology*, and *mathematics*, the latter in a specific theme called “algorithmics and programming”, and the programming language is Scratch. In high school, computer science is an independent subject with dedicated teachers, but notably, despite this, mathematics has kept the specific theme “algorithmics and programming” with Python as the programming language.

*Table 1. Synthesis of the presence of “coding” in French curricula, in mathematics and computer science.*

Grades	School level	Organization/division of School subjects	Language
1-5	Primary school	Mathematics / Computer Science (one teacher for all subjects)	Scratch Jr Robots
6-9	Middle school	Computer Science (distributed into)	Mathematics (“algorithmics and programming”)
			Technology
10-12	High school	Computer Science (“major”)	Python & others
		Mathematics (including “Algorithmics and programming”)	Python

Although France has an ambitious curriculum, there are important differences between the prescribed, implemented, and attained curricula.

The implementing of the curricula depends on:

- the reality of the material and temporal contexts of teaching,
- the variety of the experiences of teachers with coding, and the weak in-service teacher education developed for computer science,
- The balance and links that are needed to make between managing balance and links between Computer Science and Mathematics learning contents.

In practice, “attained curricula”, we can observe.

- a heterogeneity of the teachers’ practices (including not teaching coding at all),
- difficulties to achieve the learning objectives of this curriculum (at the end of primary school, the end of middle school, and the end of high school),
- an important heterogeneity of experiences and knowledge in coding for students at the end of the (mandatory) scholarship.

For the following and the example presented, we will focus on mathematics in middle school. In the mathematics curriculum for grades 7 to 9 (called “cycle 4”), the 5<sup>th</sup> theme (among 5) concerns “algorithmics and programming.” Table 2 reproduces the content of the theme, where elements are highlighted that serve to indicate the ‘thrust’ of the curriculum.

*Table 2. The part of mathematics curriculum of cycle 4 (grades 7-9) concerning “Algorithmics and Programming.”*

<p><b>Theme E - Algorithmics and Programming</b></p> <p>In Cycle 4, students are introduced to programming by developing a few simple programs in a project-based approach, without aiming for expert and exhaustive knowledge of a particular</p>
--

language or software. By creating a program, they develop programming methods, revisit the notions of variables and functions in a different form, and practice reasoning.

Examples of possible activities: games in a maze, Pong, Battleship, Nim, Tic Tac Toe, Exquisite Corpse.

### End of cycle expectations

- Write, develop and execute a simple program.

Write, develop and execute a program

#### Knowledge

- notions of algorithm and program.
- notion of computer variable.
- triggering of an action by an event.
- sequences of instructions, loops, conditional instructions.

#### Associated competences

- write, develop (test, correct) and execute a program in response to a given problem

### Example: An “Ordinary” Classroom of Mathematics in Grade 9 (15 years old)

Simon then introduced the example studied for this presentation. The mathematics teacher, Damien, is an ex-engineer with a background in programming and computer science. He places a specific focus on coding, “independently of mathematics contents” and based on video-games design. One constraint is that his teaching must take place in a computer room (figure 1), where there are not enough computers for his 9<sup>th</sup> grade class, and it is only possible to teach a half-group. So, Damian separates the class into two groups: one group works autonomously on mathematics exercises in the center of the classroom, while the other working in pairs explore pre-filled Scratch project files, containing instructions as to what they must do along with some help (figure 2).



*Figure 1. Configuration of the classroom, with computers around the room and tables in the center.*

The sequence is organized into 6 activities (planned to take over 3 or 4 one-hour-lessons and homework), with the goal of making students design a videogame as a final project:

1. Moving a car using coordinates
2. Moving the car with the arrows keys
3. Make a “realistic” U-turn
4. Manage random events
5. Manage obstacles with an associated point/scoring system
6. Make the background move

The video presented and the lesson observed concerned activity 2: making a car move using the keyboard. This activity has 3 steps. The first is to make the car move with the keyboard, which as seen in figure 3 (using an infinite loop, as expected from the teacher’s instructions). Then, the students are asked to control the speed of the car and make it fixed at 200 pixels per second. And finally, students have to make the car accelerate to 400 pixels per second, while the “d” key is pressed.

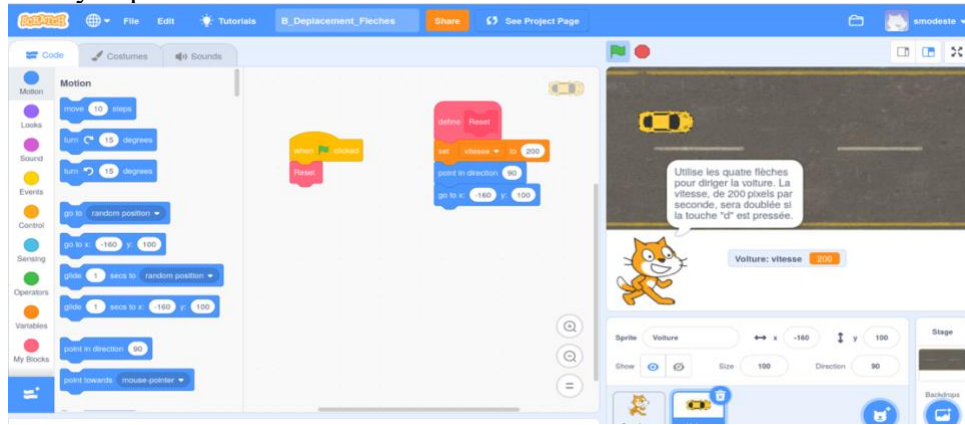


Figure 2. Example of a Scratch project, with pre-filled programs to be completed, and a tutorial programmed by the teacher with the Scratch cat presenting instructions and giving explanations and help.

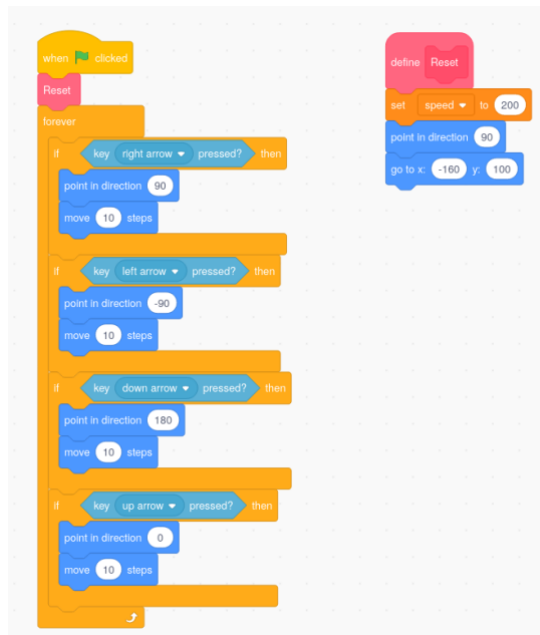


Figure 3. Example of program making the car move in the four directions using the keyboard. The students are expected to make the car move with a speed set at 200 pixels/second, and then it double to 400, while the “d” key is pressed.

The edited video was presented to show two girls working on the task and interacting with the teacher. In the video, the girls had produced a solution as in figure 3 but struggled to set the speed at 200 pixels per second. Seeing that many students had similar difficulties, the teacher gave a collective presentation at the board, where he explained the principle of building a “frame



by frame” move of the car (“like in the movies”) where the speed can be controlled. Notably, mathematical contents ‘appeared’ at this moment, because proportionality is involved through the property “speed = distance / time” and calculating distance should be completed after each step of 1/10 seconds. When the two girls come back to work, they tried to simply copy the formula given by the teacher into their program without trying to understand it. They copied “add 1/10 to speed” (speed was a variable) instead of “wait 1/10 seconds”. They then observed that it did not produce what they expected: they ran the program and observed that the car did not move quicker, although the displayed value of the speed variable increased continually. After many trials and discussions with the teacher, they were able to debug their program. They finally managed to have a correct instruction for moving at a given speed and placed it in the right place in their program. Similarly, after trial, errors, and interventions of the teacher, they also solved the last task, namely doubling the speed.

### **Conclusion: Some Observations and Topics for Discussion, From This Video**

About syntax and semantics:

- We see the important role, in coding, of the retroactions of the software in fostering an experimental approach.
- But we see also the risk of purely syntactic work from the students (kind of “push-button” effect, copying solutions...), so we ask ourselves “what did they really understand”?

About the didactical contract:

- It seems that it was not clear for these students what was expected? Was it a (formal) code, or a solution to a problem? And how is the solution validated? By who?
- We can notice the important role of the teacher, and his interactions with the students. Even if sometimes, the teacher took charge, the corrections of the program were left to the students.

Finally, about mathematics and coding:

- We presented here an interesting example of mathematics that appeared *in* and *for* coding, in particular with the speed management, which is different from the classical view of coding *for* mathematics.
- But we can observe the difficulty for students to leave the computer and its short action-feedback loop, to go back to paper-and-pencil work while coding. We see that this is difficult for the two students observed. However, it seems necessary to encourage them to do this, and to think about what they need to solve mathematically regarding their coding problem.

### **Practice-Focused Discussion Panel on Mathematics and Computing: A View from Australia.**

*Dr Elena Prieto, School of Education, University of Newcastle, Australia  
elena.prieto@newcastle.edu.au*

In September 2015, the national Australian Curriculum was officially endorsed. This curriculum includes the new Digital Technologies (DT) subject, within the Technologies learning area, which is focused on the teaching of “*computational thinking and information systems to define, design and implement digital solutions.*” (ACARA, 2018b). The document stated that this subject was to be mandatory for all Australian students from Kindergarten to Year 8 and available as an elective for Year 9 and 10 students. However, the landscape of Australian schooling is decentralized and complex, and even though the Australian curriculum places





computational thinking within the Digital Technologies curriculum, in New South Wales, its most populous state, it appears within the Science and Technology syllabus (NSW Education Standards and Assessment Authority, 2017).

Interestingly, neither of these syllabi incorporate computational thinking within mathematics, even though computational thinking is closely tied to mathematical thinking (Selby & Woollard, 2013). Additionally, very few teachers, particularly in the primary school setting, have had any formal schooling on computational thinking or coding and there were concerns that they possessed pedagogies to teach them authentically.

Presumably to facilitate incorporating the skills and content, the NSW Education Standards Authority (NESA) prepared the *Coding and computational thinking across the curriculum guide* for teachers, which aimed to develop algorithmic and computational thinking skills to better enable students and teachers to reach a coding goal. The guide highlighted the areas where computational thinking can be applied within the existing NSW K–8 syllabuses and contained activities and links to resources organized by stages of learning and learning areas. This guide has, however, been discontinued and can no longer be found in the NESA website as the new priorities have come in focusing on Aboriginal and Torres Strait Islander histories and cultures, Asia and Australia's engagement with Asia, and Sustainability.

In this climate, my work has focused on the integration of coding and computational thinking into the mathematics curriculum. In particular, over the past few years I have been working with Celia Hoyles and Richard Noss on the implementation in Australia of ScratchMaths. ScratchMaths is a two-year computing and mathematics-based curriculum for Key Stage 2 in the UK (equivalent to Years 4 and 5 in Australia).

For the Australian implementation, we conducted professional development with teachers for roughly 8 weeks, commencing with a 2-day professional development workshop and ending with a final showcase where teachers shared their experiences and samples of students' work. We also offered support during the interim classroom implementation period. The aim of the project was to explore participant teachers' perceptions of their ability to facilitate students' learning processes to develop mathematical ideas through coding, and how those perceptions varied after eight weeks of professional learning.

The project was one of the most successful ones I have run: teachers loved the materials, gained confidence in teaching coding (statistically significant!) and also **learned** coding. Furthermore, their students also showed significant improvement in their learning of coding and computational thinking but also in their enjoyment of mathematics. This research formed the pilot base of one of my PhD student's theses which proved to be a great success as signified in his appointment into the NSW Department of Education as a researcher.

### **Using Computer Programming to Bring Variables to Life in Grades 3-4 in Ontario, Canada**

*George Gadanidis, Western University*

The Ontario mathematics curriculum, grades 1-9, integrates computer programming with mathematics, in algebra and also across the other strands.

The concept of variable is in the curriculum starting in grade 1

- Grade 1: identify quantities that can change and quantities that always remain the same in real-life contexts.
- Grade 2: identify when symbols are being used as variables and describe how they are being used.

- Grade 3: describe how variables are used and use them in various contexts as appropriate.
  - Grade 4: identify and use symbols as variables in expressions and equations.
- In several grades 3-4 classrooms, we have introduced variables through coding puzzles.

This is adapted from Gadanidis, G. (2022). *Coding + Mathematics: Lessons learned from research classrooms*. <https://learnx.ca/lessons-learned>

1. Run the code below. What does it do? How does it do it?
2. Edit the code to make it draw the spiral shown on the right.

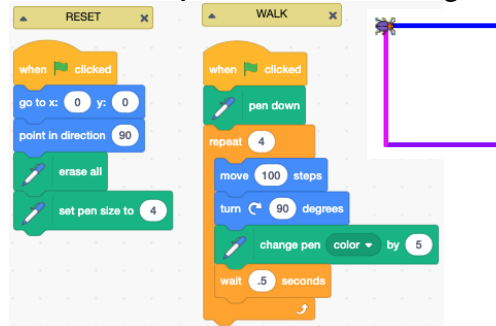


Figure 4. Code adapted from Gadanidis, G. (2022). *Coding + Mathematics: Lessons learned from research classrooms* (<https://learnx.ca/lessons-learned>), and a solution goal.

Students are able to solve the puzzle using the method of the partial code shown in figure 5 below.

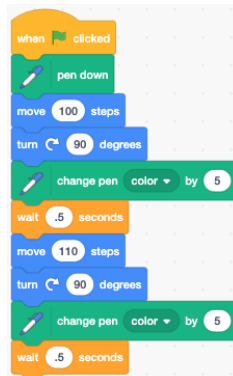


Figure 5. Students may use this method of partial code to solve the puzzle in Figure 4.

When we then ask, “Can you also solve the puzzle using a repeat block?” they are confused.

How is it possible to repeat something that needs to change?

Our approach is to then give them the code that solves this puzzle, and then present them with a new related puzzle (figure 6).

1. I found this code that seems to solve the puzzle.
2. Run the code. What does it do? How does it do it?
3. Edit the code to draw the spiral patterns shown on the right.
2. What other spiral patterns could you draw?

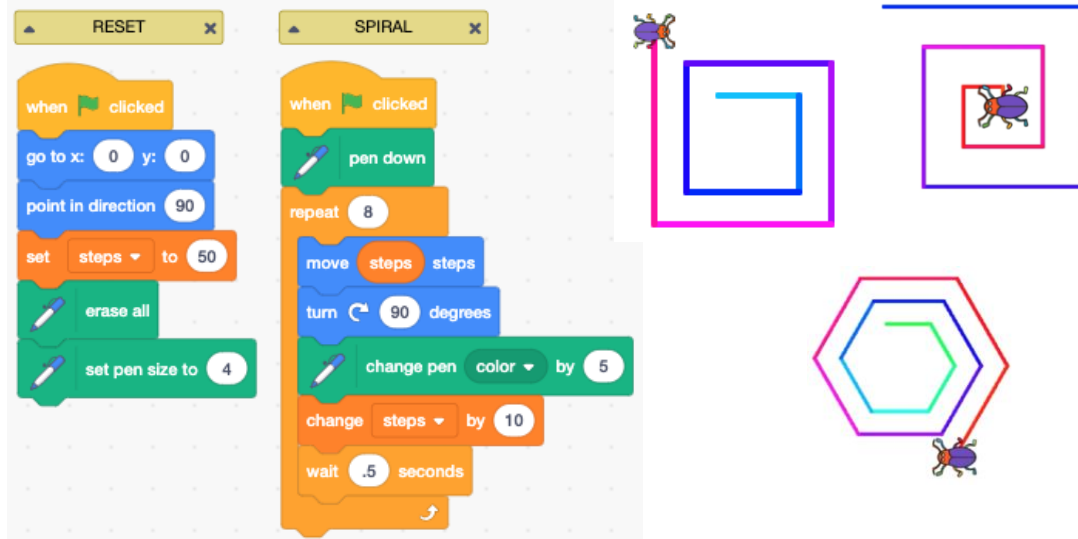


Figure 6. Code to solve previous puzzle, as well as new related puzzle and a solution goal.

Notice that we don't explicitly "teach" children how to code. They learn it incidentally as they run and edit code that works to solve related puzzles.

Coding can revitalize mathematics:

The integration of computer programming as a modelling tool may reform and revitalize mathematics education, by remediating and reorganizing mathematical concepts and relationships and consequently changing what and how mathematics is taught, and who can learn it and when. (diSessa, 2018)

... if applied to deeper mathematics:

Students – humans – thirst for deep connections, for such vistas, to have the space to flex their imagination, to wonder, to be surprised, to learn within the beauty that surrounds them, at least occasionally. (Gadanidis, Borba, Hughes & Lacerda, 2016)

## References

- diSessa, A. A. (2018). Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education. *Mathematical Thinking and Learning*, 20(1), 3-31.
- Gadanidis, G. (2022). Coding + Mathematics: Lessons learned from research classrooms. Available at <https://learnx.ca/lessons-learned>
- Gadanidis, G., Borba, M., Hughes, J. and Lacerda, H. (2016). Designing aesthetic experiences for young mathematicians: A model for mathematics education reform. *International Journal for Research in Mathematics Education*, 6(2), 225-244.

## Reaction

*Richard Noss*

In his reaction, Richard noted that this had been one of the most interesting conferences of this type and the community deserved 'a pat on the back' for all the progress they had made in both theory and practice. It seemed to him that the use of computers in mathematics was no longer questioned; it is almost taken for granted so teachers can grab the autonomy that was on offer. He mentioned the proof of the four-colour theorem (Appel and Haken in 1976), which states that no more than four colours are required to colour the regions of any map so that no two adjacent regions have the same colour) and how first proof with computers provoked considerable antagonism – the computer did nothing creative, simply served as a counting



device. The presentations in the panel pointed to more creative ways for computing, programming to interact with mathematics in ‘everyday’ classrooms.

## Summary and Reflections

*Celia Hoyles*

The panel exceeded my expectations, with 65 participants from 10 countries. This probably reflects the dramatic uptake of programming in Mathematics and also the increasing awareness of critically important complexities and possible inequities. I recommend readers watch the recording of the session to better capture all that went on. The recording can be found here: <https://cpmath.ca/symposium-proceedings/> or directly here: <https://www.youtube.com/watch?v=8YCMSvs010M&t=1s>

All the participants seek equitable outcomes in practically meaningful ways, where the partnership and interactions of coding and mathematics enriches both. I pointed to the potential of a dynamic mathematics where students have autonomy over their work and its evaluation but also noted the diverse ways coding has been introduced into the school systems. In mathematics there is always resistance to change: I recalled PME 10 In London when in his keynote Seymour Papert demonstrated an elegant proof of the circle theorems. This was greeted with delight by some (Richard and me to name but two) but horror by others who rejected this disruption of the school mathematics representational infrastructure.

The panellists took different approaches to their task. On Nam presented the history of coding and then AI in the Korean school curriculum with fascinating illustrations from words textbooks along with a video of high school geometry class. She also reported notably that the Ministry provided 60 hours of teacher training for this new curriculum.

Simon presented a fascinating ‘generic’ example of how computer science was introduced in France referring to a framework of prescribed. Implemented and attained curriculum. He noted the heterogeneity of practice, the expertise required of the teachers to help the students and the challenge of the transition between paper and pencil and computer work,

George shared some wonderful activities devised for ‘Maths with coding’ where code was given to the students and they were invited to try it, puzzle over it, explain the outcomes, and make changes as they saw fit (spiralling out rather than in for example) with some multimodal activity with students acting out ideas in the classroom. He referred to coding as a potential literacy in the sense of diiSessa and his four Rs: Reformulate, Revitalise, Remediate and Reorganise. He suggested he had found this a fruitful way to analyse the potential of tasks with the proviso that the mathematics was ‘deep’ and meaningful.

Elena talked about the history of coding and computational thinking in Australia and in New South Wales (NSW), the province where she lives; rather surprisingly not the same in turns out. In NSW, coding appears in science rather surprisingly for us from other countries. She referred to a large teaching guide produced by the state with links to a range of illustrative activities. She herself had led a project implementing ScratchMaths (the project directed by Celia and Richard in English schools, see for example Benton, L., Kalas, I., Saunders, P., Hoyles, C., & Noss, R. (2018) Beyond Jam Sandwiches and Cups of Tea: An Exploration of Primary Pupils’ Algorithm-Evaluation Strategies *J of Computer Assisted Learning* <https://doi.org/10.1111/jcal.12266>)

Elena reported significant improvements in coding and in computational thinking along with enhanced enjoyment in maths the classroom. However, she noted that there was a problem with teacher expertise both these areas. We were also rather shocked when she mentioned that



‘everything had disappeared, the manual for example as new priorities have superseded coding

...

Celia mentioned she had written a long time ago about how she had believed the computer would catalyse change but had been proved wrong – the system took over, a ‘Trojan mouse’ phenomenon sadly. See’ Hoyles, C., (1993) *Microworlds/Schoolworlds: The transformation of an innovation*. In Keitel, C., Ruthven, K. (eds) *Learning from Computers: Mathematics Education and Technology*. NATO ASI, Series F: Computer and Systems Sciences, 121, 1-17.

A more recent example of this phenomenon is that in England where computing is compulsory once unplugged activities were recognised as important planned to complement hands-on work, unplugged activities have taken over in practice and students rarely have the chance to actually interact with computers - -it saves time and hassle for schools -- but of course misses the point.

In the final discussion, concerns were voiced about the digital divide, and I pick out a few points that I found personally of particular interest. Simon mentioned that IPADS were the preferred ‘digital technology’ in France and of course it is hard to code with them so the technology actually ‘dictates’ what happens in classrooms. The issue of gender participation in Computer Science was raised. It seems that in every country represented on the panel female participation in CS was a major concern. Elena also suggested that when computers were given to schools for a particular purpose, they in fact were rarely used for that purpose. On a more positive note, George mentioned how teaching coding in mathematics led to ‘seeing something new in students’; it brought a fresh lens on to who engaged with the subject and how. This is certainly something that Richard and Celia have found over the years and regard it as a fundamentally important way to combat fixed hierarchies as to who is ‘good and ‘bad’ at the subject (see for example Noss, R. and Hoyles, C. (1996) *Windows on Mathematical Meanings: Learning Cultures and Computers*. Dordrecht: Kluwer Academic Publishers)

Other challenges to change were mentioned such as high stakes testing, the proliferation of resources but it was reiterated that we need a *long*-term view, learning from the past while planning for the future. We in the research community can only plant seeds to stimulate and support critical learners. Let us maintain our vision of the outstandingly and unique positive possibilities of this work: the high motivation, the sense of student agency and bring the joy back to learning mathematics for all students.





## Using Coding to Enhance Numeracy Teaching and Learning

Andrijana Burazin<sup>1</sup>, Taras Gula<sup>2</sup>, Miroslav Lovric<sup>3</sup>

<sup>1</sup>University of Toronto, Mississauga; <sup>2</sup>George Brown College;

<sup>3</sup>McMaster University, [lovric@mcmaster.ca](mailto:lovric@mcmaster.ca)

The purpose of this paper is to suggest coding as a means of working on *numeracy tasks*, which, in turn, can be used to enrich mathematics instruction. We believe that the non-traditional approaches and intuitive reasoning that are natural components of a numeracy task should establish their place in solving problems in mathematics.

What is numeracy? There is no consensus on the definition of numeracy (Geiger, Goos & Forgasz, 2015), and its relation to quantitative literacy, quantitative reasoning, and mathematical literacy remains ill defined. As well, mathematics-related terms, such as number sense, sense making, mathematics in context, and even modelling, are sometimes used to characterize or describe numeracy. The view that numeracy and mathematics are two different entities has been gaining attention and influence. Steen (1997) writes: “Numeracy is not the same as mathematics, nor is it an alternative to mathematics. Today's students need both mathematics and numeracy. Whereas mathematics asks students to rise above context, quantitative literacy is anchored in real data that reflect engagement with life's diverse contexts and situations.”

Perhaps the most illustrative way of conceptualizing numeracy is to view it as living in the space between the school mathematics and the mathematics that is used in real-life situations. The transfer from the abstract (as done in a math classroom) to the concrete (math in the real world), once thought to be automatic and unproblematic, turned out to be anything but. This transfer is the key defining feature of numeracy; in other words, numeracy is about linking concrete quantitative situations to abstract mathematics (specifically arithmetic); see Gula & Lovric (2023) for a discussion about numeracy and its development.

Rather than trying to say what numeracy *is*, by presenting case studies of numeracy tasks, we discuss what numeracy is *about*. Hence, our focus, and this contribution, are on the teaching of numeracy. Teaching numeracy at various levels of education - from elementary, to tertiary, to adult - remains a challenge that has been addressed by (in our view, still incomplete) research and debate. Geiger, Goos & Dole (2011) write: “far less is known about how teachers [of numeracy] learn about, appropriate and then create effective mathematics teaching practices.”

What is a numeracy task? Unlike a mathematics task, which is about mathematical ideas and objects and their relations (and which lives in the abstract), a numeracy task is about concrete, context-centred phenomena (using the abstract to make sense of the concrete). It is different from a typical word problem in mathematics: although a math word problem might sound like it is about the real world, deep down it is about some math concept and often requires a particular mathematical technique (see the first example in our discussion). A numeracy task is about a problem involving quantities (numbers) that a real person would indeed meet in their professional or personal life, and whose solution is obtained using the means that the actual person in that situation would, or might, use. In Gula & Lovric (2023) and Burazin, Gula & Lovric (2023) we conceptualize numeracy tasks and discuss ways of judging whether a given task is a good numeracy task. Borrowing from the Wolfram's (2016) computational thinking model, we view a numeracy task as having four essential components: define – abstract – compute – interpret and communicate.

We start by analyzing a sample mathematics textbook word problem, to serve as a contrast to numeracy tasks that we discuss.

### A Math Textbook Word Problem (And a Bit More)

The following mathematical task, or its variations, can be found in every calculus textbook:

*Find the dimensions of a rectangle with perimeter 100 m whose area is as large as possible.*

This particular version is from Stewart (2021; page 332); sometimes, the problem is presented as a word problem, for instance about a farmer who has a fixed amount of fencing, and wants to fence off the largest possible rectangular piece of land. (Note: although this might sound like a real life situation, it is certainly not – when buying a piece of land, are we ever told: “Here is a rope, whatever you can enclose with it is yours.”)

As this problem appears in the optimization section in the textbook, students will, most likely, solve it by writing equations. Denoting the length of the field by  $l$  and its width by  $w$ , the perimeter satisfies  $2l + 2w = 100$ , and one has to maximize the area  $A = lw$ . By combining the equations  $A$  is reduced to a function one variable, and students will follow the usual (expected!) strategy: differentiate and find critical points. Although it will likely yield a correct answer, this method does not give any insights about why the answer makes sense.

An alternative approach is to experiment: assume that the length is  $30m$ , and the width is  $20m$  (so that the perimeter is  $100m$ ). The area is  $20 \cdot 30 = 600m^2$ . If we increase the length, say to  $35m$ , and decrease the width to  $15m$  (keeping the perimeter at  $100m$ ), the area is  $525m^2$ . So that does not work. Let’s try to make the longer side a bit shorter: if we decrease the length to  $28m$  and increase the width to  $22m$ , the area is  $28 \cdot 22 = 616m^2$ , which is larger than the original area! If we further decrease the longer side, say, to  $27m$ , then the shorter side is  $23m$ , and the area further increases to  $621m^2$ . So it seems that if we take away from the longer side and give to a shorter side, we can increase the area. We can do this as long as there *is* a longer side! Thus, when both sides are equal (and equal to  $25m$  in this case), we get the largest area of  $625m^2$ . The answer is the most symmetric of all rectangles – the square.

Now let’s use coding. This short code in Python

```

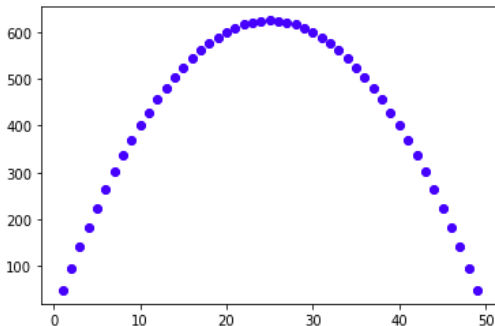
1 length=0.0
2 print(" length", " width", " area")
3
4 while length<50:
5     width=50-length # 2length+2width=100
6     area=length*width
7     print(f"{length:8.3f}{'':2}{width:8.3f}{'':2}{area:9.3f}")
8     plt.scatter(length,area,color='b')
9     length+=1

```

runs, in a loop, the length of the rectangle from 0 to 50, calculates the corresponding width and the area, and prints in a table, parts of which are below:

length	width	area
0.000	50.000	0.000
1.000	49.000	49.000
2.000	48.000	96.000
3.000	47.000	141.000
4.000	46.000	184.000
5.000	45.000	225.000
...		
22.000	28.000	616.000
23.000	27.000	621.000
24.000	26.000	624.000
25.000	25.000	625.000
26.000	24.000	624.000
27.000	23.000	621.000
28.000	22.000	616.000
29.000	21.000	609.000
...		

In our case, the length starts at 0 and is incremented by 1 (of course, the increment can be changed to non-integer values). As well, the code produces a graph of the area as a function of the length, which shows an obvious symmetry (with additional lines of code we could annotate the graph, change the domain of the function, etc.):



This example illustrates that even a mundane math word problem can be repackaged into a rich experience. Non-traditional approaches (informal, or not), typical for numeracy tasks (as we will see shortly), stimulate creativity and enhance understanding. This is our point: numeracy tasks and approaches can enhance and enrich mathematics teaching and learning.

In the remaining part of this paper we discuss several numeracy tasks where one can use coding. To make the presentation shorter, we avoid giving a full, detailed real-life context. As in the symposium workshop, a reader is encouraged to look at the code, copy into a Python notebook, run it, and then modify and experiment.

#### Four Case Studies of Numeracy Tasks

**Example 1** Good decision, or not? (Modified from a post on the Personal Finance subreddit).

*There was a good opportunity, so I bought a \$6000 leather couch on a sale for \$5000. I paid with my visa, because I had no cash to spend. I can afford about \$200 a month to pay off my credit card debt. Am I being reasonable?*

Of course, there is a formula somewhere that could answer this question. But that's not the point (and, quite likely, the person in this story would not be able to find it and use it). Here is the approach we suggest: we will follow what happens to the \$5000 over time, neglecting all other charges that could have been made on the credit card. From the cardholder agreement or a monthly statement we learn that the annual percentage rate (APR) is 19.99%.

The cyclic nature of the problem:

balance at the start of the month → payment of \$200, assumed to be done on the first day of the month → interest accrued over the month → balance at the end of the month, equal to the balance at the start of the next month

suggests algorithmic approach, so this is a native coding problem.

The first step in approaching a numeracy task is to abstract it, i.e., strip of all inessential information. Hence the assumption that the payment is applied on the first day of the month, and (to simplify code) taking a month to have 30 days, and a year to have 365 days. (Challenge: modify the code to eliminate these assumptions.) From the Cardholder Agreement we read:

The amount of interest we charge you on each account statement is calculated as follows:

- first, we determine your average daily balance by adding the interest-bearing amount you owe each day and dividing that total by the number of days in the statement period
- next, we determine the daily interest rate by dividing the annual interest rate by the number of days in a year.

Your interest charge is then calculated by multiplying the average daily balance by the daily interest rate by the number of days in a statement period.

Now we have all information we need to code:

```

1 apr=0.199
2 balance=5000
3 monthly_payment=200
4 month=0
5 total_interest=0
6
7 while balance>0:
8     month+=1
9     balance=balance-monthly_payment
10    interest=(apr/365)*30*balance
11    if balance>0:
12        total_interest+=interest
13        balance+=interest
14
15 print("Debt paid after",month,"months.",f" Total interest paid={total_interest:6.2f}")

```

Debt paid after 32 months. Total interest paid=1345.38

Using a while loop we are able to control the iterative process, so that it stops when the balance reaches zero. Because of our simplifying assumption, the balance is the same for all days in a given month, and hence it is the average daily balance. The calculation shows that the total interest paid (\$1345.38) exceeds the sale difference (\$1000); so in reality, the person will have to pay more than the full price for the couch. As well, it will take 32 months to bring the balance (for that purchase only!) to zero.

By adding a print statement, we obtain a listing which shows the dynamics of the interest accrued and the remaining balance:

month= 1	start balance= 4800.00	interest= 78.51	end balance=4878.51
month= 2	start balance= 4678.51	interest= 76.52	end balance=4755.03
month= 3	start balance= 4555.03	interest= 74.50	end balance=4629.53
month= 4	start balance= 4429.53	interest= 72.45	end balance=4501.99
month= 5	start balance= 4301.99	interest= 70.36	end balance=4372.35
month= 6	start balance= 4172.35	interest= 68.24	end balance=4240.59
month= 7	start balance= 4040.59	interest= 66.09	end balance=4106.68
month= 8	start balance= 3906.68	interest= 63.90	end balance=3970.58
month= 9	start balance= 3770.58	interest= 61.67	end balance=3832.25

Having a functioning code means that we can experiment. For instance, by changing the monthly payment to \$500, we realize that the total interest the person would pay is a bit over \$400, and the balance will be paid off fully in 11 months.

```

1 apr=0.199
2 balance=5000
3 monthly_payment=500
4 month=0
5 total_interest=0
6
7 while balance>0:
8     month+=1
9     balance=balance-monthly_payment
10    interest=(apr/365)*30*balance
11    if balance>0:
12        total_interest+=interest
13        balance+=interest
14
15 print("Debt paid after",month,"months.",f" Total interest paid={total_interest:6.2f}")

```

Debt paid after 11 months. Total interest paid=408.10

**Example 2** Simulating rare events (communicated by a colleague physician; to protect privacy, we do not identify the source, nor the actual disease or the school board in question)

*In the case of an identified epidemic in a school, school district board administrators need to decide on the course of action, which might include closing all district schools. On average, every year 2 students in a large district school board in Ontario are diagnosed with a medical condition (we call it MC), which could cause an epidemic (such as meningitis). In the first eleven months in 2022, four children in that board have been diagnosed with MC. What should school administrators do? Declare an epidemic and close schools, or keep them open? The school board in question has about 200,000 students. The known incidence of MC is 1 in 100,000 per year.*

The number of students diagnosed with MC changes from year to year: there are years when no students are diagnosed, and there could be years when 2 or 3, or more students are diagnosed with MC.

No one can know for sure whether or not an epidemic is about to break. Math cannot help us answer that question either. But what can it do? A good question to ask is: what is the chance (i.e., how likely is it) that 4, or more, students being diagnosed with MC in one year happened just by chance, rather than being due to an actual epidemic?

We model this situation using probability. One simulation (lines 9-11 in the code below) checks each student every day of the year for the presence of MC (in order to speed up calculations, we used the units of thousands; hence 200 represents 200,000 students). It does so by picking a random number between 0 and  $100 \cdot 365 - 1$ , thus implementing the probability  $1/100 \cdot 365$  that a randomly selected student has been diagnosed with MC on a given day (if the randomly selected number is 0, the student has MC, otherwise not).

This loop is then run 500 times (so performing 500 repeated experiments), and the number of times the number of students with MC equals, or exceeds, 4 is recorded.

In this particular run of the code, 4 or more cases occurred 71 times, and so we conclude that 4 or more cases of MC occurred by pure chance (i.e., not due to the epidemic) is  $71/500$ , or 14.2%.

```

1 num_students=200 # units are thousands
2 day_value=100*365 # incidence 1/100*365/day; need to account for the avg of 2
3 num_simulations=500 # number of simulations
4 count_4ormore=0 # number of simulations that ended with 4 or more cases
5
6 for k in range (num_simulations):
7     meningitis_cases=0
8     for i in range(365):
9         for j in range(num_students):
10            if random.randrange(0,day_value)==0: # positive case identified
11                meningitis_cases+=1
12            if meningitis_cases>=4:
13                count_4ormore+=1
14
15 print("In",num_simulations,"simulations, 4 or more cases occurred",count_4ormore,"time(s)")
16 print("The chance of 4 or more cases occurring is",count_4ormore/num_simulations)

```

```

In 500 simulations, 4 or more cases occurred 71 time(s)
The chance of 4 or more cases occurring is 0.142

```

This is how far quantitative thinking goes. Now, with this useful piece of information, it is up to the school administrators and health care professionals to decide on the course of action.

Note: Using statistics, we can model this situation with a Poisson distribution with the parameter  $\lambda = 2$ . In that case, we compute the chance to be

$$1 - \frac{e^{-2} \cdot 2^0}{0!} - \frac{e^{-2} \cdot 2^1}{1!} - \frac{e^{-2} \cdot 2^2}{2!} - \frac{e^{-2} \cdot 2^3}{3!} = 0.14288$$

(see Lovric (2012) for details) i.e., a about 14.3%. Thus, our simulation was pretty close! (Note: the second time we run it, we obtained 14.8%).

**Example 3** Understanding the Mean (What to do when a TA is late grading their papers?)

*There are four sections in a large university course. The course instructor, about to start their lecture, wishes to discuss the test that the students wrote two weeks ago. However, one of their TAs has not finished marking. Based on the 300 marked tests, the instructor calculated the mean to be 67.6%. Rather than waiting for the TA to finish marking the remaining 100 tests, the instructor decides to report the average of 67.6% for the entire class. Find an argument to support their decision.*

Obviously, the issue is the following: if we know something about a sample (i.e., a subset) of a population, what (if anything) can we say about the population itself?



Suppose that a population consists of five numbers, of which we know four: 5, 6, 8, and 17 (hence, this is a sample). Is there anything we can say, with some certainty, about the mean of the population of all five numbers, if the mean of this sample is 9? If the unknown number is 1, the mean is  $37/5=5.4$ . If the unknown number is 9, the mean is 9, and if the unknown number is 27, the mean is  $63/5=12.6$ . So, the mean of the five numbers (population mean) could be (lot) smaller, equal or (lot) larger than the mean (sample mean) of the known four numbers, and we cannot claim anything, with certainty, about the relation between the two means.

But in this case, the sample of 300 is much larger than 4. Would that change anything? To figure it out, we write a Python code to generate 400 random test grades (assumed to be normally distributed with a mean of 67 and a standard deviation of 30) and store them in a list (if the grade randomly generated is smaller than zero, we change it to zero). Then we calculate the means of the first 300 grades in the list, and compare it with the mean of all 400 grades. Here is the code:

```

1 import numpy as np
2 import numpy.random as nrand
3 from statistics import mean
4
5 grades=np.round(nrand.normal(67,30,400))
6 for i in range(len(grades)):
7     if grades[i]<0:
8         grades[i]=0
9 print(grades)
10
11 print("The average grade of 300 tests is",mean(grades[:300]))
12 print("The average grade of all 400 tests is",mean(grades[:400]))

```

To illustrate, we show a part of the list that contains students' test grades, and then the desired output:

```

[ 91.  75.  57.  98.  13.  63.  65.  33.  83.
  32. 138.  79.  53. 103.  90. 101.  25. 110.
 129.  40.  96. 116.  36.   0.  83.  52.  70.
  56.  56.  85.  38.  52.  62.  49.  97.  58.
  67.  29.  54.  62. 127.  88.  84.  56.  65.
  29.  17.  69.   7.  98.  51.  93. 108. 110.
  39.  14.  91.  69.   0.  61.  44.  22.  98.]
The average grade of 300 tests is 67.596666666666666
The average grade of all 400 tests is 67.98

```

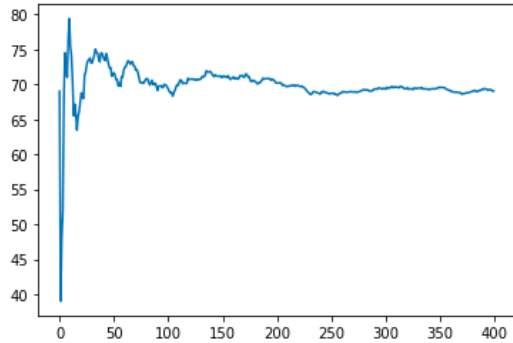
So, the two averages are close to each other. Running the code over and over again, we obtain different means, but in all cases the sample average (300 grades) is close to the population average (all 400 grades). In conclusion, the course instructor can report, with confidence, the average of 300 tests as being close to the entire class average.

The fact that this is so is not a coincidence, but a reflection an important fact from statistics. To get a bit better feel about it, we ask the following question: How does the class mean behave if we compute the mean of the first two tests in the list, then the mean of the first three tests, then the mean of the first 4 tests, and so on? Again, a code will help us figure it out: we generate a list of 400 random grades, and then, in a loop compute the means:

```

1 import numpy as np
2 import numpy.random as nrand
3 import matplotlib.pyplot as plt
4 %matplotlib inline
5 from statistics import mean
6
7 grades=np.round(nrand.normal(67,30,400))
8 for i in range(len(grades)):
9     if grades[i]<0:
10        grades[i]=0
11 print(grades)
12
13 averages=[]
14 for i in range(400):
15     averages.append(mean(grades[:i+1]))
16
17 plt.plot(averages)
18 plt.show()

```



Here are the first few grades from the list:

```
[ 69.  9.  66.  63. 135. 105.  64.  57. 119.
```

and their averages below: 69 is the average of the first grade, 69; the next term is  $(69+9)/2=39$ , followed by  $(69+9+66)/3=48$ ,  $(69+9+66+63)/4=51.75$ , and so on.

```
1 print(averages)
[69.0, 39.0, 48.0, 51.75, 68.4, 74.5, 73.0,
```

The plot (the number of test grades on the horizontal axis and the corresponding mean on the vertical axis) reveals what is going on: after initial large fluctuations, the means stabilize, and approach the mean of the entire population:

In conclusion, if a sample is large enough, its mean is a good approximation of the mean of the entire population (in statistics, this is called the Law of Large Numbers).

**Example 4** Understanding growth: size and volume (or, how mixing up the meaning of the two lead to a major error in a research paper)

*In the article “Is clinical breast examination an acceptable alternative to mammographic screening?” published in the British Medical Journal (Mittra et al, 2000), we read:*

*[...] if you consider the exponential growth rate and doubling time of breast cancer you find that a single breast cancer cell has to undergo 30 doublings to reach a size of 1 cm, when it will contain  $10^9$  cells and be clinically palpable. Since the average size of a non-palpable, mammographically detected cancer can be assumed to be about 0.5 cm, the lead time gained by mammography over clinical breast examination would be of the order of only one doubling. Whether this lead time equivalent of one doubling in the natural course of 30 doublings would lead to a significantly greater reduction in mortality is questionable.*

*Find a major flaw in this reasoning, thus alerting the physicians not to undervalue the significance of the lead time gained by mammography.*

Note that the term *size* in the article refers to a linear dimension (units are cm). Doubling time is the time needed for cells to double in number, and hence to double the volume. Mixing these two concepts is what lead to the error in this paper.

We write code to understand the exponential dynamics of the cell growth better. We assume (as is standard in practice) that a single cell generates two offspring cells, and that the cells accumulate in the form of a sphere. In the code we track the generation, i.e., count the doublings, and record the number of cells, the length and the volume of the cancer:

```

1 nocells=1
2 length=1/1024 #size (diameter) of a cell in cm
3
4 print("generation"," number of cells"," length (cm)"," volume (cm^3)")
5
6 for generation in range(1,31):
7     nocells*=2
8     length*=2**(1/3)
9     volume=4*np.pi*(length/2)**3/3 #assume spherical
10    print(f"{generation:5}{nocells:21}\t{length:.7f}\t{volume:0.10f}")

```

The starting lines of the output are:

generation	number of cells	length (cm)	volume (cm <sup>3</sup> )
1	2	0.0012304	0.000000010
2	4	0.0015502	0.000000020
3	8	0.0019531	0.000000039
4	16	0.0024608	0.000000078
5	32	0.0031004	0.000000156
6	64	0.0039062	0.000000312
7	128	0.0049216	0.000000624
8	256	0.0062008	0.000001248
9	512	0.0078125	0.000002497
10	1024	0.0098431	0.000004993

and the last few lines, relevant for this discussion, are here:

25	33554432	0.3149803	0.0163624617
26	67108864	0.3968503	0.0327249235
27	134217728	0.5000000	0.0654498469
28	268435456	0.6299605	0.1308996939
29	536870912	0.7937005	0.2617993878
30	1073741824	1.0000000	0.5235987756

The number of cells in the 30<sup>th</sup> generation (doubling) is a bit over one billion, and the size is 1cm (so, the claim in the paper “...30 doublings to reach a size of 1 cm, when it will contain 10<sup>9</sup> cells...” is correct). Note that from one generation (one doubling) to the next, the volume doubles, but the length does not - the length increases by a factor of  $\sqrt[3]{2}$ .

So, for the length to double from 0.5cm to 1 cm, it takes three doublings, not one (thus the claim “...the lead time gained by mammography over clinical breast examination would be of the order of only one doubling” is incorrect).

The error in the paper is quite serious; some health authorities and administrators tried to argue, based on the claim about the lead time of one doubling, that a mammography (which is expensive) might not provide a benefit of early cancer detection compared to a less costly clinical breast examination.

### Conclusion

In this paper we presented examples which illustrate how certain numeracy tasks can be solved using coding. In the conceptualization of a numeracy task based on Wolfram’s computational model define – abstract – compute – interpret and communicate, coding is part of *compute* stage. Since they appear in authentic, real life contexts (rather than being artificial as many word problems in math textbooks are), numeracy tasks, besides having the value on their own, could enrich teaching mathematics. As well, the non-traditional approaches and intuitive reasoning that are natural components of a numeracy task should establish their place in solving problems in mathematics.



## References

- Burazin A., Gula, T. & Lovric, M. (2023). Numeracy course “Numbers for Life” at McMaster University: Research into the effectiveness of the course instruction, and quality numeracy tasks. In: TBA (Eds.). *Proceedings of the Adults Learning Mathematics (ALM 30) Conference, 3-5 July 2023, Limerick, Ireland*. (in preparation)
- Geiger, V., Goos, M., & Dole, S. (2011). Teacher Professional Learning in Numeracy: Trajectories Through a Model for Numeracy in the 21st Century. In J. Clark, et al. (Eds.), *Mathematics: Traditions and [New] Practices. Proceedings of the 34th annual conference of the Mathematics Education Research Group of Australasia and the Australian Association of Mathematics Teachers*. Adelaide: AAMT and MERGA.
- Geiger, V., Goos, M., & Forgasz, H. (2015). A rich interpretation of numeracy for the 21st century: A survey of the state of the field. *ZDM*, 47, 531-548.
- Gula, T. & Lovric, M. (2023). Numeracy tasks: provoking transfer between concrete and abstract thinking spaces. Submitted to *Numeracy*.
- Lovric, M. (2012). *Calculus for the Life Sciences: Modelling the Dynamics of Life. Probability and Statistics*. Toronto: Nelson.
- Mitra I., Baum, N., Thornton, H., & Houghton H. (2000). Is clinical breast examination an acceptable alternative to mammographic screening? *BMJ* 2000; 321: 1071-1073. doi: 10.1136/bmj.321.7268.1071.
- Steen, L. A. (Ed.). (1997). *Why numbers count: quantitative literacy for tomorrow's America*. New York: College Entrance Examination Board.
- Stewart J., Clegg, D. K. & Watson, S. (2021). *Calculus: Early Transcendentals, 9th Edition*. Brooks Cole.
- Wolfram, C. (2016). Anchoring Computational Thinking in today’s curriculum. Available at: <https://www.conradwolfram.com/writings/anchoring-computational-thinking-in-todays-curriculum>. Accessed 19 July 2023. See also <https://www.computationalthinking.org>



## Coding and Computational Thinking: A Time that Came and Went but Will Always Be Ours

Elena Prieto-Rodriguez  
University of Newcastle

In September 2015, the national Australian Curriculum was officially endorsed. This new curriculum included a Digital Technologies (DT) subject, within the Technologies learning area, which was to be focussed on the teaching of “*computational thinking and information systems to define, design and implement digital solutions*” (ACARA, 2018a, 2018b). The document stated that this subject was to be mandatory for all Australian students from Kindergarten to Year 8 and available as an elective for Year 9 and 10 students. All the different States and Territories in the country set to task and by 2018, education authorities had incorporated coding and computational thinking into their respective syllabi. Interestingly, neither of these syllabi incorporated computational thinking within mathematics, even though computational thinking is closely tied to mathematical thinking (Selby & Woollard, 2013). Also, at the time, there were serious concerns about the fact that very few teachers, particularly in the primary school sector, had any formal training in either of these knowledge areas, and there were also concerns that they possessed pedagogies to teach them authentically (Falkner et al., 2014).

Many Professional Development (PD) initiatives for teachers were thus developed by universities and private organisations to address concerns related to teacher preparation for the Australian DT subject (Commonwealth of Australia, 2016). These initiatives ranged from Massive Online Open Courses (MOOCs), such as those run by the University of Adelaide’s Computer Science Education Research (CSER) group (Vivian et al., 2014), and face-to-face workshops (Bower et al., 2017; Chalmers, 2018). A great effort was made to link these PD opportunities to the different syllabi, so that teachers would find the translation from PD to classroom use as seamless as possible.

In 2018, we wrote a paper where we quoted a colleague stating that by adding the terms ‘coding’ or ‘computational thinking’ to a research grant proposal in the field of education, you would pretty much guaranteed funding (Hickmott & Prieto-Rodriguez, 2018). While this was said in jest, there was a feeling at the time in Constructionist circles, that the teachings of Seymour Papert were now ripe for worldwide implementation. Indeed, Grover and Pea called computational thinking ‘a competency whose time has come’ (2018, p.1), and claimed:

“In a world infused with computing, computational thinking is now being recognized as a foundational competency for being an informed citizen and being successful in all STEM work, and one that also bears the potential as a means for creative problem solving and innovating in all other disciplines” (2018, p. 34).

Presumably to facilitate incorporating knowledge of coding into all these other disciplines, the NSW Education Standards Authority (NESA) prepared the *Coding and computational thinking across the curriculum guide* for teachers. This guide aimed to develop algorithmic and computational thinking skills to better enable students and teachers to reach a coding goal. The guide highlighted the areas where computational thinking can be applied within the existing NSW K–8 syllabuses and contained activities and links to resources organised by stages of learning and learning areas.



Five years and a global pandemic have passed and, at least in Australia, the professional development funding climate seems to have lost interest in coding and computational thinking in favour of the old classics of basic literacy and numeracy. No longer we see an appetite for innovation that would see young people develop into the creative thinkers of the future, but we are forced into standardised lesson plans and strategies to improve our PISA rankings based on rote learning and memorisation. The *Coding and computational thinking across the curriculum* guide has been discontinued and can no longer be found in the NESA website. When I started writing this paper before the symposium in 2022 there were new priorities that teachers should focus on for integration across the curriculum: Aboriginal and Torres Strait Islander histories and cultures, Asia and Australia's engagement with Asia, and Sustainability.

The much-needed training for teachers to implement the digital technologies syllabus has also become harder to provide. Teachers have very busy schedules and having our PD initiatives accredited with authorities to contribute to the 50 hours of PD required of teachers, was always a way to ensure participation. This has also changed. It is practically impossible to have our initiatives accredited unless we have staff devoted to preparing endless applications every time we offer a workshop.

And yet, in this climate, my work continues to focus, as it did twenty years ago, on the integration of coding and computational thinking into the mathematics curriculum. Whether it is a funding priority, a curriculum priority; whether it is in the syllabus or not, I still believe in the power of computing to support the learning of mathematics and vice versa.

I am not the only one. From the 1960s, a limited yet highly influential group of educational researchers has delved into the integration of computer programming to enhance the understanding of mathematics. The year 2006 marked a turning point with the popularisation of the term 'computational thinking' by Jeannette Wing, triggering a notable upsurge in research activity within this field. The body of literature that connects mathematics education with computational thinking is not insignificant. In a systematic analysis we conducted in 2017, at the beginning of the explosion of funding mentioned above, we found that a substantial portion of studies originated from computer science academics rather than experts in the field of education. We also noted that although mathematics is somewhat a focus of the research, studies tend to revolve around teaching programming skills. Additionally, a predominant portion of these studies adopted small-scale research designs focused on self-reported attitudes and beliefs. As a result, we drew the conclusion that there are significant opportunities to pursue more robust research designs that explicitly target mathematics and report on tangible learning outcomes (Hickmott et al., 2018).

But first we need to be clear on what these “tangible outcomes” are. If they are defined by achievement in formal examinations, we need to ask ourselves: will results in a standardised mathematics test taken by children who learned coding tell us whether they have indeed mastered any mathematics content that is worth knowing?

And as for the research designs, how do we know that the teachers who we train to facilitate this new way of knowledge construction, have indeed learned something? In terms of this second question, in 2018 we reported on our six year-long experience training teachers using Seymour Papert's Constructionist principles (Papert, 1980). Our research concluded that testing teachers to ascertain whether they had learned the concepts and skills we were teaching them really did detract from the experience (Hickmott & Prieto, 2018).

As for the first, over the past few years I have been working with Celia Hoyles and Richard Noss on the implementation in Australia of ScratchMaths. ScratchMaths is a two-year computing

and mathematics-based curriculum for Key Stage 2 in the UK (equivalent to Years 4 and 5 in Australia). For the Australian implementation, we conducted professional development with teachers for roughly 8 weeks, commencing with a 2-day professional development workshop and ending with a final showcase where teachers shared their experiences and samples of students' work. We also offered support during the interim classroom implementation period. The aim of the project was to explore participant teachers' perceptions of their ability to facilitate students' learning processes to develop mathematical ideas through coding, and how those perceptions varied after eight weeks of professional learning.

The project was one of the most successful ones I have run: teachers loved the materials, gained confidence in teaching coding (statistically significant!) and also learned coding. Furthermore, their students also showed significant improvement in their learning of coding and computational thinking but also in their enjoyment of mathematics.

We observed increases in participants' teaching self-efficacy across both mathematics and computing (see Table 1). A paired-samples t-test was conducted to check for differences before and after attending the workshop and after attending showed a number of statistically significant results. There was a change in teachers' perceptions of their ability to teach mathematics with programming before ( $M = 3.3$ ,  $SD = 0.47$ ) and after the intervention ( $M = 4.5$ ,  $SD = 0.08$ );  $t = -5.09$ ,  $p = 0.037$ . There was also a statistically significant difference in their self-efficacy with regards to coding and computational thinking before ( $M = 3.2$ ,  $SD = 0.78$ ) and after the intervention ( $M = 4.3$ ,  $SD = 0.23$ );  $t = -5.05$ ,  $p = 0.002$ .

*Table 1. Pre- and post-survey results (items in Mathematics scale preceded by an asterisk).*

	Pre	Post	Gain
I feel confident using simple programs for the computer.	4.60	4.71	0.11
I know how to teach programming concepts effectively.	2.67	4.14	1.48
I can promote a positive attitude towards programming in my students.	4.13	4.57	0.44
I can guide students in using programming as a tool while we explore other topics.	2.93	4.43	1.50
*I can guide students in using mathematical thinking as a tool when programming.	3.13	4.43	1.30
I feel confident using programming as an instructional tool within my classroom.	2.67	4.17	1.50
I can adapt lesson plans to incorporate programming as an instructional tool.	2.93	4.29	1.35
I can create original lesson plans, which incorporate programming as an instructional tool.	2.87	4.14	1.28
I understand how mathematics concepts relate to programming concepts.	3.00	4.43	1.43
I appreciate the value of teaching mathematics and programming in an integrated manner.	3.87	4.57	0.70

\*All items were presented in a 5-point scale from Strongly Disagree (coded as 1) to Strongly Agree (coded as 5).

It was to hear about the positive outcomes and insights from the ScratchMaths professional development workshops and the subsequent trial period. The teachers' feedback provided valuable information about the effectiveness of the program and areas for improvement. Here's a summary of the key points from the research project:



1. Positive Reception and Student Engagement:
  - Teachers expressed positive sentiments about ScratchMaths after the trial period.
  - Student work samples and reflective comments showcased students' engagement in learning.
  - Teachers reported that students looked forward to ScratchMaths sessions each week.
  - The resources were praised for being well scaffolded and promoting collaboration and social support for learning.
2. Increased Self-Efficacy:
  - Participant teachers reported an increase in their self-efficacy with mathematics and coding.
  - One teacher mentioned that ScratchMaths made complex concepts, like 2D shapes, practical and understandable.
  - Coding provided an opportunity for teachers to feel supported in an area where they might not have felt as confident (e.g., coding for a teacher strong in mathematics).
3. Integration of Mathematics and Coding:
  - While ScratchMaths led to sustained student engagement, not all students engaged equally with the mathematical concepts within the activities.
  - Teachers acknowledged that the mathematical aspects needed to be more explicitly directed to ensure all students engaged with them.
  - Some students used a trial-and-error approach to complete activities, rather than engaging with the mathematical concepts.
  - Teachers acknowledged that the activities were effective in reinforcing concepts already taught in a practical manner.
4. Differing Perspectives on Learning Outcomes:
  - In one regional focus group, teachers viewed coding as a more significant learning outcome than mathematics.
  - In the metropolitan area, teachers observed that some students focused more on trial-and-error approaches rather than mathematical problem-solving.
  - All teachers agreed that the activities were useful for reinforcing concepts.

Based on this feedback, we concluded that ScratchMaths was successful in engaging students and improving teachers' self-efficacy. However, there's still a need to address the varying levels of engagement with the mathematical content among students. Future iterations of the program could focus on providing more explicit guidance for students to engage with the mathematical aspects of the activities. Additionally, it might be valuable to continue emphasising the integration of mathematics and coding, highlighting how coding can serve as a tool for enhancing mathematical understanding. This could involve refining the instructional approach to strike a better balance between coding exploration and mathematical engagement.

Overall, the feedback from the teachers who participated provided important insights for refining ScratchMaths to align with the desired learning outcomes and ensuring that both coding and mathematics are effectively integrated. These results were presented in the annual STEM conference (Prieto-Rodriguez et al., 2019).

We have conducted another four years of (heavily interrupted by a worldwide pandemic) workshops for teachers since the results above. Our amalgamated analysis of survey responses



collected during these 4 years of professional development shows results entirely consistent with the ones presented in the 2019 publications.

As a final reflection I would like to remark that this research shows that both teachers and students benefit from the integration of mathematics and computer science, and will continue to do so, whether funding agencies consider it worthwhile or not.

### References

- ACARA. (2018a). Digital Technologies curriculum rationale. Retrieved from <https://www.australiancurriculum.edu.au/f-10-curriculum/technologies/digital-technologies/rationale/>
- ACARA. (2018b). F-10 Curriculum - General Capabilities. Retrieved from <https://www.australiancurriculum.edu.au/f-10-curriculum/general-capabilities/>
- Bower, M., Wood, L. N., Lai, J. W., Highfield, K., Veal, J., Howe, C., ... & Mason, R. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education*, 42(3), 53-72.
- Chalmers, C. (2018). Robotics and computational thinking in primary school. *International Journal of Child-Computer Interaction*, 17, 93-100.
- Commonwealth of Australia. (2016). *STEM Programme Index*. Retrieved from <http://www.chiefscientist.gov.au/2016/01/spi-2016-stem-programme-index-2016-2/>
- Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. *Computer science education: Perspectives on teaching and learning in school*, 19(1), 19-38.
- Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2018). A scoping review of studies on computational thinking in K–12 mathematics classrooms. *Digital Experiences in Mathematics Education*, 4, 48-69.
- Hickmott, D., & Prieto-Rodriguez, E. (2018). To assess or not to assess: Tensions negotiated in six years of teaching teachers about computational thinking. *Informatics in Education*, 17(2), 229-244.
- Falkner, K., Vivian, R., & Falkner, N. (2014). *The Australian Digital Technologies Curriculum: Challenge and Opportunity*. Paper presented at the Australasian Computing Education, Auckland, New Zealand.
- Papert, S. A. (1980). *Mindstorms: Children, Computers, And Powerful Ideas*. Hachette UK.
- Prieto-Rodriguez E, Holmes K, Hickmott D, Berger N, (2019) 'Using Coding to Teach Mathematics: Results of a Pilot Project', Using Coding to Teach Mathematics: Results of a Pilot Project, Queensland University of Technology, Brisbane, Australia.
- Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition. Paper presented at the 18th annual conference on innovation and technology in computer science education, Canterbury, United Kingdom.
- Vivian, R., Falkner, K., & Falkner, N. (2014). Addressing the challenges of a new digital technologies curriculum: MOOCs as a scalable solution for teacher professional development. *Research in Learning Technology*, 22.



## Re-Designing Coding-Based Mathematics Tasks into Scaffolded and Project Formats: Two Pre-Service Teachers' Perspectives

Samantha Boerkamp, Jessica Sardella, with Chantal Buteau  
Brock University

### Introduction

During the Coding, Computational Modelling, and Equity in Mathematics Education Symposium, various mathematics/programming activities were presented. These included those from the PD Day as well as the Working Groups, Keynotes, Poster Presentations, and Panel Discussions. The activities were designed with varying levels of coding and mathematics knowledge in mind, whereby some have also included additional foci, such as equity. The structure and guidance provided for each activity also differed depending on the purpose of its presentation— i.e. to present real, useable math coding activities, to prompt discussion around creation of math coding activities, to discuss implementation of coding in the math classroom, etc.

Within Working Group B: Coding and Computational Modeling in Secondary and University Mathematics Education, the leaders proposed a task for the members to work on throughout their sessions. This task involved selecting and working with a provided coding-based mathematics activity to modify it for a certain group of students. Following the symposium event, we (the two first authors) took on the task that was suggested. For the purpose of this paper, we selected two activities that incorporated coding in different ways, re-designed them in a) a step-wise guided activity ('scaffolded'), and b) a project-based activity, and reflected on our process. Below, we outline the context of our work, including our background, the approach we took, and a brief overview of our process including our selection of activities. We then present the activities as they were provided to us at the symposium and selected aspects of our engagement when working through them, followed by our scaffolded and project versions, and highlights of key changes we made and why. We conclude with some final thoughts comparing our process of task re-design for each activity.

### Our Math + Coding Experience at Brock University

As Mathematics/Education majors in Brock University's Concurrent Education program with the intention of becoming math teachers, we have had experience with coding-based mathematics activities. These experiences mostly stem from our completion of a sequence of three programming/math courses, called *Mathematics Integrated with Computers and Applications* (MICA). Consisting of a total of 14 projects, these courses engage students (such as ourselves) in programming-based math investigation projects in which they design, program, and use interactive computer environments to investigate mathematics concepts, conjectures, theorems, or real-world situations (Buteau et al., 2015). The final MICA course is split into two sections: MICA III for math/stats majors, and MICA III\* for future teachers such as ourselves. MICA III\* provides students with explicit insight into the affordances of programming for math and supports preservice understanding of teaching in relation to the new coding curriculum in Ontario (Ontario Ministry of Education, 2020; 2021). This is done through guided reflections, comparisons between coding languages, and a final project in which students collaborate with in-service teachers to prepare and implement a coding math activity in the classroom.





It is with our experiences in the MICA courses and as pre-service teachers that we engaged in the two task re-design and reflected on our decision-making and thought processes when adapting the activities.

### **Our Task Design Approach: Two Versions**

After the Symposium, we took the opportunity to look further into two of the presented activities and adapt them to create both a 'scaffolded' and 'project' version of each. The scaffolded version of these activities guides the student throughout their learning, giving explicit instructions and explanations so the student can achieve the desired result and gain the appropriate understanding of the concepts (Waite & Grover, 2020). While still structured, the project versions of these activities are more independent and allow the student to apply the concepts while demonstrating the skills and knowledge they have acquired with more opportunity for decision making and problem solving. Along with the activity guidelines, we also included a brief "Teacher's Guide" with each activity outlining necessary prior knowledge and helpful instructional tips. By creating two versions of each activity, we illustrate different ways that programming can be integrated into the mathematics classroom for different purposes or needs.

### **Selection of the Two Coding-Based Mathematics Activities and Overview of Our Process**

The first activity we selected was a 'Cryptography' activity from Working Group B that Jessica (and Chantal) had begun to work with during the Working Group sessions. This activity focused on cryptography methods related to the Caesar cipher and coding with a key. It was provided with some code and base questions in a separate document, with the aim of modifying it to be a suitable activity for a certain audience, e.g. secondary/first-year university students. As presented, we felt it had both a focus on the mathematics and coding aspects, as students would need to understand the mathematics to adapt the code. For this activity, we initially both created our own versions of a scaffolded activity, but due to their similarities we decided to combine them into one version encompassing both of our ideas. We each created our own project versions of the Cryptography activities.

The second activity was presented by Callysto at the PD Day and focused on the probability of flipping a coin with an emphasis on the Law of Large Numbers. This activity was provided in a complete form, ready to implement in a classroom. While it included code, we felt the focus was more on the mathematics learning from *using* the code, rather than the code (or parts of the code) itself. For this activity, we again had both initially created our own versions of a scaffolded activity. This time we both took very different approaches, with Jessica focusing only on modifying the provided activity, and Sam focusing on adding an extension to the task. As we both felt our activities were incomplete, we found combining these two approaches help to create a more complete activity for students' learning. Unlike the 'Cryptography' activity, for this activity we created the project version after combining our scaffolded versions. Because of this, we both had similar ideas of what the project would look like, resulting in only one version.

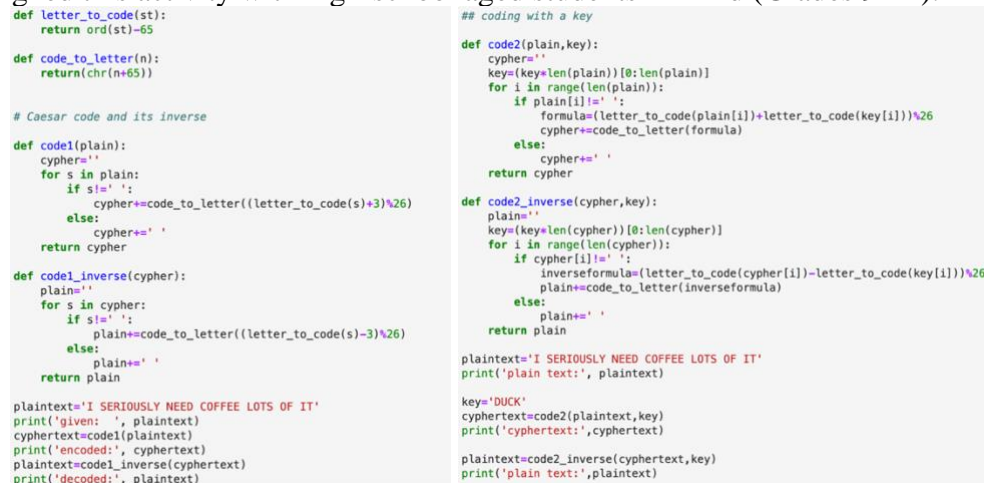
#### **'Cryptography' Activity**

The 'Cryptography' activity provided by Working Group B included an activity guide (particularly useful for the instructor) outlining the programming language, suggested starting level for both math and programming, mathematical and computational concepts, and thinking/habits of mind, as well as brief instructions. The activity involves an exploration of basic coding and decoding techniques such as translation (known as the Caesar code) and coding

with a key which is based on arithmetic modulo 26. The full code was also provided for both cryptography methods.

- See the activity guide here: [Cryptography Activity Guide](#)
- See the provided code here: [Cryptography Provided Code](#)

When initially looking at this activity, we were both somewhat overwhelmed as it was a lot of code provided at once and including some syntax we did not know. To overcome this, we each decided to run through the code first to see what it outputted. After seeing the output, we both tried to work through each line of the code to try and understand how the output was formed. Essentially, we were breaking down the code into smaller chunks to see what each section did- something that we decided to incorporate into our ‘re-designed’ activities (see ‘Scaffolded ‘Cryptography’ Activity’). We both also had to do some of our own research on the Caesar code and coding with a key to better understand what the code (and math involved) was aiming to do, as these were new concepts to us. Sam also tried altering some of the code to try and understand why some of the components were there, including altering the shift in the Caesar code and removing the ‘%26’ to understand why it was used. Jessica researched more about the coding concepts, including what ‘+=’ was used for, which helped in understanding how words could be encoded. Coding concepts we both had to research included the *ord* and *chr* functions, as neither of us had seen this used before. Overall, after doing some basic research, neither the math nor the coding was overly complicated. Because of this, we feel that this activity could be suitable for beginner coders (especially the scaffolded version) or more advanced coders, depending on the code provided. Based on the math and coding involved in this activity, we re-designed this activity with high-school aged students in mind (Grades 9-12).



```

def letter_to_code(st):
    return ord(st)-65

def code_to_letter(n):
    return(chr(n+65))

# Caesar code and its inverse

def code1(plain):
    cypher=""
    for s in plain:
        if s!=' ':
            cypher+=code_to_letter((letter_to_code(s)+3)%26)
        else:
            cypher+=" "
    return cypher

def code1_inverse(cypher):
    plain=""
    for s in cypher:
        if s!=' ':
            plain+=code_to_letter((letter_to_code(s)-3)%26)
        else:
            plain+=" "
    return plain

plaintext="I SERIOUSLY NEED COFFEE LOTS OF IT"
print('given: ', plaintext)
cyphertext=code1(plaintext)
print('encoded:', cyphertext)
plaintext=code1_inverse(cyphertext)
print('decoded:', plaintext)

## coding with a key

def code2(plain,key):
    cypher=""
    key=(key*len(plain))[0:len(plain)]
    for i in range(len(plain)):
        if plain[i]!=' ':
            formula=(letter_to_code(plain[i])+letter_to_code(key[i]))%26
            cypher+=code_to_letter(formula)
        else:
            cypher+=" "
    return cypher

def code2_inverse(cypher,key):
    plain=""
    key=(key*len(cypher))[0:len(cypher)]
    for i in range(len(cypher)):
        if cypher[i]!=' ':
            inverseformula=(letter_to_code(cypher[i])-letter_to_code(key[i]))%26
            plain+=code_to_letter(inverseformula)
        else:
            plain+=" "
    return plain

plaintext="I SERIOUSLY NEED COFFEE LOTS OF IT"
print('plain text:', plaintext)

key="DUCK"
cyphertext=code2(plaintext,key)
print('cyphertext:', cyphertext)

plaintext=code2_inverse(cyphertext,key)
print('plain text:', plaintext)

```

Figure 1. Left: Code provided by the Working Group leaders for encrypting and decrypting for the “Caesar Cipher” portion of the ‘Cryptography’ activity. Right: Code provided by the Working Group leaders for the “Coding with a Key” portion of the ‘Cryptography’ activity.

### Scaffolded ‘Cryptography’ Activity

- Student Copy: [Cryptography Scaffolded- Student Copy](#)
- Solution Guide: [Cryptography Scaffolded- Solution Guide](#)

Following conversation with other Working Group B members, we wanted to first add some more context behind the mathematical content being explored through this activity. Adding a brief explanation of what the Caesar cipher is and/or how a Cipher Wheel works aims to give students a general understanding of how their program should work and what their goal actually

is. We also added an introduction to modular arithmetic, as this is likely a new concept for students and is necessary in order to shift letters past Z and back to the start of the alphabet.

The main, and arguably most important, change that was made to this activity was breaking it down into smaller steps, and adding explanations. As mentioned, when going through the activity on our own, we were at times overwhelmed and struggled to understand what parts of the code meant. It took us significant time to go line by line and understand what was happening. For beginner coders, we felt like this would be too much all at once. We also added “TRY IT!” tasks to the worksheet to encourage incremental coding and working, in order to help encourage students to understand each aspect before they all come together.

In terms of what is given to the students, we decided to give students the functions to convert between letters and numbers. Python’s `ord()` and `chr()` commands were new to us and would also likely be new to students. Providing this code does not take away from students’ math learning as it is more of a technical aspect of the coding that they can be introduced to and learn for future use. This also meant introducing the Unicode as it is what the functions are based on and will help students to see why they will need to add or subtract 65 to the functions for easier use. Students can then alter and explore the codes and may find ‘limitations’ with them, e.g., that this can only be used for individual characters and not words or sentences.

We also decided to guide students through the process of encrypting, but leave the decrypting process to them to work through. Starting with encoding a letter, moving to a word, and then to a sentence helps familiarize the students with the concept of the cipher (shifting) and the syntax (e.g. ‘+=’), while still requiring them to think computationally (through the use of loops, if/else statements, etc.). As some code is provided, students are guided to alter this code for certain goals, and then asked to create their own decrypting function, following a sort of “Use, Modify, Create” model (Waite & Grover, 2020).

For the coding with a key section, we first encouraged students to try this method on paper so they can see how it works before trying to code it. This is also something they use later on when verifying if their code works. Regarding the code, a ‘fill-in-the-blank’ encryption code is provided to students to encourage them to figure out how to apply the math to the code using their knowledge from the first part of the activity (Caesar cipher) as well as other methods (e.g. trial and error, etc.). Again, students are asked to create the decryption code on their own since this is very similar to that for encryption, before being asked to explore the encryption method further— this time with regards to the security of the key.



Figure 2. Left: A section of the scaffolded ‘Cryptography’ activity in which students are guided through programming the encryption process of the Caesar cipher. Students should discover that the provided code does not consider spaces when encrypting and that this will affect the decrypting process. Students are then asked to alter the code to fix this problem. Right: An example of a solution code to encrypt a message.



## Jessica's 'Cryptography' Project

- Link here: [Jessica's Cryptography Project](#)

When designing my version of the Cryptography project, I relied on my scaffolded version in order to ensure that the main outcomes of the program would be similar. I also referenced some of the MICA course assignments in order to come to a sort of structure for the assignment—having students engage in the mathematical investigation process ‘as a mathematician would’, starting with research, engaging in the programming and debugging process, and finishing with a sort of discussion on the topic (Buteau et al., 2023). Although this is a project, and less structured in terms of the process, I wanted to ensure clear guidelines on what is expected; since they are being provided a topic, I think it’s important they know the goal. By breaking the guidelines for the program into different points, my goal is to encourage students to work incrementally, and to help with the organization of the program.

I decided to provide students with the functions to convert between numbers and letters (for the same reason as in the scaffolded activity), and the code to encrypt a word since beginner coders likely wouldn’t know how to move from one letter to the next (i.e. using +=). Since this is a project, I think that providing only this code is reasonable. Students may work in a similar process to that in the scaffolded activity, but they will need to think of these steps and encounter the various challenges (e.g. needing to account for spaces between words) on their own before resolving them.

## Samantha's 'Cryptography' Project

- Link Here: [Sam's Cryptography Project](#)

When designing the project, I knew it was going to be much less scaffolded than the activities made before. I only wanted to introduce just enough of the mathematics so that the students would understand the concept on paper, but not too much where they wouldn’t be able to investigate it on their own. This is why I only gave a brief definition of each encryption method.

I am also assuming that the students would have already been introduced to cryptography and its purpose, as a project is usually considered for an end of unit activity or summary so the students would not need such an elaborate introduction to the concept in order to succeed. While an activity is made for them to build their learning through guided steps, the project is for them to explore their understanding on their own terms without step-by-step guidance telling them what to do. This allows the students to express their understanding in a way that makes sense to them, without a unique right solution. Keeping to the goal of the activity, I asked them to create their own codes representing the Caesar model and encryption with a key model but with very limited code provided to them. I justified that the students would already have a strong understanding of loops and conditionals that they would not need anything else other than what was given to them. After creating the code, I wanted them to be able to explore it, which is why I asked them to try and break another classmate’s cipher. This allows them to use their code with a purpose while also using their math knowledge on the subject to try and decipher another student’s message. It is an application of how ciphers are used in the real world. The next part of the project makes them reflect on these two models of cryptography by asking how they could make these stronger and less breakable in their own new form of ciphering text.

I found creating the project more difficult than the scaffolded activity because it was something I was not familiar with doing. I was unsure at what level of student I should be creating this activity for or how much work they should be doing. My goal was to try and



incorporate all of the activity's components into the project, just without the scaffolding that was in place for the activity. This meant asking them to do things that incorporated all of the ideas they needed to obtain by the end of it. It also meant giving them more opportunity to explore on their own and creating questions that would encourage them to do so. A key moment was when I decided to include the partner work in this project, because it gave the opportunity for exploration of these models without direct guidance from the teacher of how to do so. The basis of cryptography is sending secret messages between two people, so it made sense to include a partner for them to test their knowledge and capabilities with. After this, it was easier for me to understand what I wanted the students to achieve from this project, and how a project differs from an activity.

### Flipping Coins' Activity

The 'Flipping lots of Coins' activity provided by Callysto included a Python notebook with pre-made code and instructions. The full code is included as a part of the activity; i.e. students are not asked to code, rather they are asked to *run* the program to explore the mathematical concepts involved. The activity involved probability, with students exploring the "Law of Large Numbers" through the simulation of flipping a coin. Programming concepts utilized in the code differed from the 'Cryptography' activity, utilizing the random function, graphing, and storing data in tables.

- See the activity provided here: [Flipping Coins Provided Code](#)

We both looked at this activity after looking at the 'Cryptography' activity and felt that it was quite different in terms of structure, format, and concepts. The code itself was relatively simple, and since students are not actually doing any of the coding, we felt that the activity was more math-based with potential for the coding aspect to be overlooked. While we still worked through the code to try and understand it, we did not do the same level of research as we did with the 'Cryptography' activity. This is because it is not necessary to understand how the code is working to understand the math, since most of the code is related to graphing. While students may not gain programming skills from this, they are able to benefit from some of the affordances of programming such as automation (i.e. through simulation of a large number of coin flips, e.g.  $N=100,000$ ). Compared to the 'Cryptography' activity, we felt that this activity was already quite scaffolded and broken down for students, as the code was already divided into individual cells that could be run separately from each other together with guidelines in separate text-cells. For a student in Grade 7-10 (the age range we had in mind when re-designing the activity), the mathematical concept itself is likely not too difficult to understand and could be considered beginner friendly.



Figure 3. Left: A section of Callysto's 'Flipping Coins' activity in which students run the cell to display the results of a given number of coin flips in a table. Right: Students can then visualize the results of the coin flips in a histogram by using the code provided.



## Scaffolded 'Flipping Coins' Activity

- Student Copy: [Flipping Coins Scaffolded- Student Copy](#)
- Solution Guide: [Flipping Coins Scaffolded- Solution Guide](#)

For the Flipping Coins task, our main goal was to engage students more, including in the programming aspect, rather than just have them run through the code. To do so, we added various questions throughout for students to think about as they go through the activity (e.g. making predictions, what they notice, forming conclusions about changing number of flips, etc.). We also redesigned the actual coding involved in the task to: a) allow students to do some of the coding on their own; b) utilize coding concepts or possible solutions that we feel encourage more computational thinking (e.g. utilizing loops and conditional statements).

For the 'biased coin' part of the task, we decided to provide students with an alternative method for coding an un-biased coin. Since this method, which may be used for a biased coin as well, is slightly more advanced than that they would create on their own we felt it would be good to provide. Students will still need to understand why this code works and modify it to fit the given situation, again using the "Use, Modify, Create" model (Waite & Grover, 2020).

We also added on to the activity, using Callysto's suggestion that this activity lead into the Monte Carlo approach. We felt that the Flipping Coins code provided enough to guide students, looking for challenge, to create their own representation of the Monte Carlo method involving rolling a dice and explore probabilities that they may not already know (i.e. probability of rolling different sums of dice). We also added an extension activity, in which students can take what they learned earlier to simulate a different dice-rolling game. Each of these parts becomes less scaffolded and allows the students to work more independently to develop their coding skills for mathematics learning.

*Lucky Number Seven:* A player rolls two dice and can only win if sum of the dice adds to 7. All other possible summations are considered a loss.

Think! How can we alter the code we used above to determine the probably of winning this game?

Response:

Try it! Fill in the code below to simulate 2 dice being rolled 10 times. Add comments throughout.

```
N=
for i in range(N):
    dice1 = random.choice([1,2,3,4,5,6])
    print("Dice 1 Roll", i+1,":", dice1)
    dice2 = random.choice([1,2,3,4,5,6])
    print("Dice 2 Roll", i+1,":", dice2)
```

Try It! Fill in the code so the sum of the dice is added after each roll. Add comments throughout.

```
N=
for i in range(N):
    dice1 = random.choice([1,2,3,4,5,6])
    dice2 = random.choice([1,2,3,4,5,6])
    sum =
    print('The sum of the dice on Roll', i+1,"is:",)
```

Figure 4. A portion of our scaffolded "Flipping Coins" activity, in which we ask students to use what they have learned to create a program that determines the probability of winning the "Lucky Number Seven" game. A fill-in-the-blank code is provided.

## Our 'Flipping Coins' Project

- Link here: [Flipping Coins Project](#)

When designing the 'Flipping Coins' project, we again used our scaffolded version as a guide for what we wanted to include. Again, we tried to keep the concepts and content very similar, but left the process of the investigation and coding more to the student. Part 1 (Task A and B) has the students creating their coin flip simulations and discovering the Law of Large Numbers and investigating experimental vs theoretical probabilities. Similar to the scaffolded version, students form predictions, create coin flip simulators, and run them a varying number of



times to compare theoretical and empirical probabilities. The main difference here is that there is very minimal code provided to students. As this is a project, we assume that they have the coding knowledge necessary to do this on their own, and want to encourage them to think computationally to figure out what that code may look like. Part 2 has students wrap up their findings and tie it all together. This includes students forming a conclusion, and creating an adapted version of their program to further validate their learning. This additional simulation also allows students to see a different tool for a probability simulation, showing them different potential outcomes while giving them the freedom to choose their own tool to model. We felt that adding this additional component was necessary to push students forward in their individual thinking since Part 1 was still broken down for them in terms of what they needed to do.

### Final Thoughts

The provided activities were both different in terms of the structure, coding concepts, and math. Thus, we found our experiences adapting them progressed differently as well. We found the Cryptography activity easier to adapt than the Flipping Coins activity. This could be because the Flipping Coins activity provided to us was already quite scaffolded with a limited total number of code lines, making us feel limited in our creativity, whereas the Cryptography activity had the code provided (including multiple lines, and control commands) but didn't provide the same kind of guidance for exploration allowing for more creative freedom. The Cryptography activity also included an activity guide (useful for the instructor) with explanations of the concepts that were being covered, making the goal for the activity clearer and providing background on what students should be gaining. The Flipping Coins activity felt limited in terms of where to go with it, although we recognize that there are applications that could be explored (such as those of the Monte Carlo method) but we were not familiar enough with them at the time of creating the activities to explore them further. We also found it useful to start with the scaffolded versions before moving on to the project design, as we had already broken down the concepts and knew what would need to be included. This helped in deciding how much guidance to provide within the project instructions.

Overall, our experience creating these scaffolded and project activities allowed us to think more deeply about different aspects involved when integrating programming with mathematics. We used our own experiences from learning programming in a similar context and as future teachers to think about what students may need to get started, as well as what may provide more of a challenge. Our experiences allowed us to get a feel for what the appropriate amount of guidance should be and helped in our decision making. We hope that we can implement these activities in classrooms in the near future and get both indirect and direct feedback on the design of our activities. From here, we could further adapt the activities to better meet the needs and skills of our students.

### References

- Buteau, C., Muller, E., & Ralph, B. (2015). Integration of programming in the undergraduate math program at Brock University. Retrieved at <http://researchideas.ca/coding/docs/ButeauMullerRalph-Coding+MathProceedings-FINAL.pdf>
- Buteau, C., Broley, L., Dreise, K., & Muller, E. (2023). Students using programming for pure and applied mathematics investigations. *Épíjournal de Didactique et Epistémologie des Mathématiques pour l'Enseignement Supérieur*. <https://doi.org/10.46298/epidemes-9190>



Bauerkamp, S., Sardella, J., & Buteau, C. (2023). Re-Designing Coding-Based Mathematics Tasks into Scaffolded and Project Formats: Two Pre-Service Teachers' Perspectives. In Online Proceedings of the *Coding, Computational Modelling, and Equity in Mathematics Education Symposium*, St. Catharines (Canada), April 2023.

Ontario Ministry of Education (2020). Elementary curriculum: Mathematics.

<https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics>

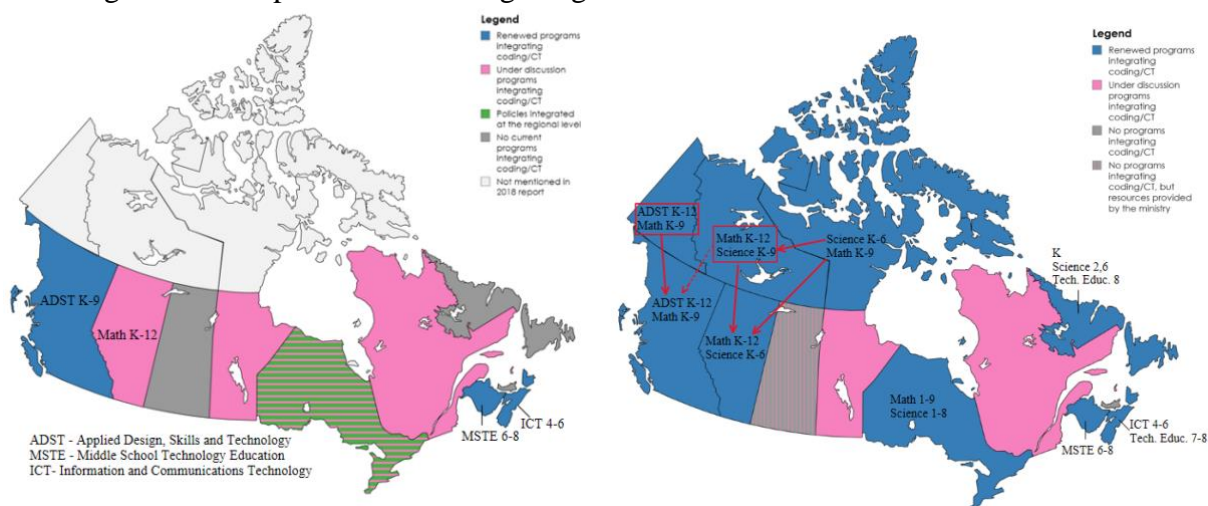
Ontario Ministry of Education (2021). Mathematics, Grade 9. <https://tinyurl.com/3djym45h>

Waite, J., & Grover, S. (2020). Worked examples & other scaffolding strategies. In S. Grover (Eds.), *Computer Science in K-12: An A-to-Z handbook on teaching programming* (240-249). California, CA: Edfinity.

## Integration of Coding and Computational Thinking in Compulsory Education: The Landscape in Canada - A Report.

Emma Molinaro, with Chantal Buteau  
Brock University  
May 2024

In 2018, a report on the integration of computational thinking, programming, and coding across the Canadian provinces was published (Gannon & Buteau, 2018). As education in Canada is determined at the provincial/territorial level, the report addressed the various levels of integration in the provinces such as having mandatory courses, elective courses, optional online resources, or no current plans of integration. This report aims to update those levels of compulsory integration, ranging from kindergarten to grade 12, to the current year (2024). This report is written with information on the curriculum from the respective Ministry of Education, and does not look at the actual implementation in the classroom. Figure 1 exemplifies an overview of the level of integration each province/territory was at in 2018 (left) versus the level of integration they currently reside at (as of April 2024). Moreover, it indicates which curriculum the coding and/or computational thinking integration occurred/occurs in.



*Figure 1. Integration of coding and/or computational thinking in school curricula across Canadian provinces/territories in 2018 (left) and 2024 (right). Solid red arrows indicate that the province/territory uses the same curriculum as the province/territory it is pointing to. Dashed red arrows indicate that the province/territory is switching to the same curriculum as the province/territory it is pointing to.*

Canada is not the only country that has been working towards the integration of computational thinking in their school curriculum. For instance, the European Union published a report in 2016 and an updated report in 2022 outlining the status of the integration of computational thinking skills in multiple European countries school curricula. Figure 2 shows two maps comparing the integration in the 2016 report versus the 2022 report.

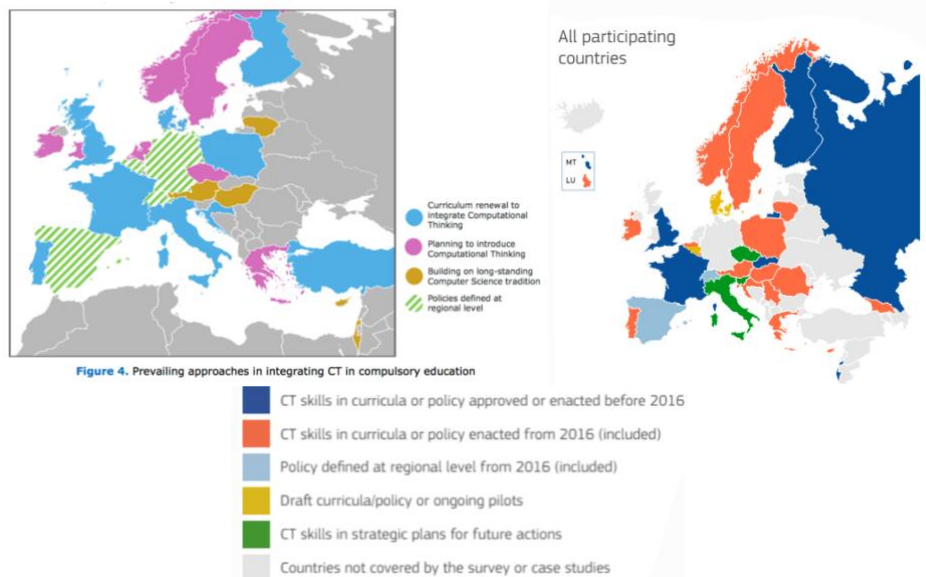


Figure 2. Integration of computational thinking (CT) skills in school curricula in Europe in 2016 (left) (Bocconi et al., 2016) versus 2022 (right) (Bocconi et al., 2022)

In addition to the European countries shown in the Figure 2, Australia, South Africa, and the United States of America, are only a few examples of other countries who have worked towards integrating computational thinking into their curriculum (Stephens et al., 2021).

### British Columbia

British Columbia has been, and continues to be at the forefront of integrating coding and computational thinking into their province wide curriculum. Since the 2016-2017 school year, a redesigned curriculum called Applied Design, Skills and Technology (ADST) was being integrated in kindergarten to grade 9, including a focus on coding in grades 6 to 9 (British Columbia News, 2016). As of the current year, 2024, this redesigned curriculum has been implemented in all grades from kindergarten to grade 12 (British Columbia Ministry of Education, n.d.). Starting with kindergarten to grade 5, the goal of the ADST curriculum is to provide exploratory and purposeful play opportunities for students to begin developing the foundational skills in design thinking and making. Then, in grades 6 and 7, students explore more specific areas of the ADST curriculum through the selection of a minimum of three modules, including computational thinking, computers and communication devices, digital literacy, and more. The courses for grades 8 and 9 are more flexible in how they are offered, as schools are required to provide a ‘full year course’, but this could be offered as one or more modules of various lengths. Lastly, in grades 10 to 12, students can choose to follow a specialized path in a specific area, such as Business Education, Information and Communications Technology, etc., or move forward following broader interests (British Columbia Ministry of Education, 2018).

In addition to the ADST curriculum, the use of technology and coding has also been integrated into the kindergarten to grade 9 mathematics curriculums in British Columbia. For example, from as early as Kindergarten, one of the Curricular Competencies under reasoning and analyzing is to use technology to explore mathematics. From kindergarten to grade 5, this technology is mainly calculators, virtual manipulatives, and concept-based apps. However, from





grade 6 and onward, coding and programming are introduced under the Curricular Competencies' logic and patterns, and modelling, respectively (British Columbia Ministry of Education, 2016).

### **Yukon Territory**

“Yukon schools follow the British Columbia (B.C.) curriculum, with adaptations to include: Yukon content; and Yukon First Nations’ ways of knowing and doing” (Government of Yukon, 2023, Para. 1). Thus, coding and computational thinking are integrated throughout the whole territory.

### **Alberta**

Alberta is another province that has been an innovator in the integration of computational thinking and coding into their curriculum. In the 2018 Gannon & Buteau report, the 2017 draft of Alberta’s new scope and sequence for kindergarten to grade 12 mathematics was explored. This draft illustrated the inclusion of computational thinking and/or coding throughout majority of the grade levels under every Essential Understanding (Alberta Education, 2017). Since then, Alberta has continued to incorporate coding and computational thinking into other curriculums, such as the science curriculum as computer science has become a topic implemented as a unit of study from kindergarten to grade 6 (new LearnAlberta, 2023). In the subject introduction, it is stated that “The study of computer science allows students to apply creativity, design, and computational thinking and to develop scientific inquiry and problem-solving skills” (new LearnAlberta, 2023, p.1). The descriptions of the science curriculum at each grade level are then broken down into three main sections; Organizing Idea, Guiding Question, and Learning Outcome. From kindergarten to grade 6, the Computer Science Organizing Idea is “Problem solving and scientific inquiry are developed through the knowledgeable application of creativity, design, and computational thinking” (p. 15). Then, for grades 3-6 we see computational thinking being more integrated in the Guiding Questions and Learning Outcomes. For instance, in grade 3 there are multiple mentions of computational thinking, as the Learning Outcome for this grade level is “Students investigate creativity and its relationship to computational thinking” (p.37). Thus, under the Learning Outcome subcategories (Knowledge, Understanding, and Skills & Procedures) computational thinking is divided into explorable aspects, such as what it includes (e.g., breaking down a task, designing instructions, etc.), how it is a process that requires the use of creativity, and more. In grades 5 and 6, computational thinking and coding are then brought together. For instances, the use of computational thinking in the design process of computational artifacts (new LearnAlberta, 2023, p. 59).

### **Northwest Territories**

In a recent CBC News article, it was discussed how Northwest Territories has followed Alberta’s education curriculum since the 1950s, but is now beginning shift toward British Columbia’s curriculum (Lachacz, 2023). Although some changes have begun, NWT currently still follows Alberta’s mathematics curriculum (Government of the Northwest Territories, n.d.-a) and Alberta’s grade 7 to 9 science curriculums. Although NWT does not follow Alberta’s kindergarten to grade 6 science curriculums, they have their own science and technology curriculum that has been implemented since 2004 (Government of the Northwest Territories, 2004). In this curriculum, the ability to use computer programs to enhance and deepen student learning and understanding is emphasized. Additionally, “students can use computer programs to compile, organize, and store data gathered through investigations; to write reports and papers in



which they present their findings...; and to work with simulations in areas of study in which hands-on activities are not feasible (e.g., in astronomy) or in which there is too great a safety risk” (Government of the Northwest Territories, 2004, p. 15). Therefore, there is a significant amount of coding integration in the NWT curriculum, and as they transition into using British Columbia’s curriculum, there will be even more.

### **Nunavut**

Nunavut’s curriculum is divided into four strands, one being Iqqaqqaukkaringniq which is an “integrated core curriculum that focuses on ways in which we describe and improve our world. Concepts in mathematics, analytical and critical thinking, solution-seeking, innovation, technology and practical arts will be explored” (Department of Education Ilinniaqtuliyikkut, n.d., p. 11). The approved curriculum falling under this strand is either unique to Nunavut or draws on the curriculums from other provinces/territories. For instance, Nunavut follows Alberta’s kindergarten to grade 9 mathematics curriculum and NWT kindergarten to grade 6 science and technology curriculum (Department of Education Ilinniaqtuliyikkut, 2019). With the inclusion of these two curriculums, Nunavut has therefore integrated coding and computational thinking into their education.

### **Saskatchewan**

Since the 2018 Gannon & Buteau report there has been no updates to the mathematics or science curriculum at any grade level in Saskatchewan. Thus, Saskatchewan currently has no formal integration of coding or computational thinking in their curriculum (Saskatoon Industry Education Council, 2024b). However, something that is being adopted and integrated in the kindergarten to grade 12 classrooms, at the teacher’s discretion, is “SaskCode” activities. SaskCode is not required to be introduced into the classroom, but it is an option that is available to all teachers in Saskatchewan which has been growing in popularity over the last few years with it being used in over 4,200 classrooms (Saskatoon Industry Education Council, 2024a). The main goal of SaskCode is to “equip teachers throughout Saskatchewan with the pedagogy, technological skills, and physical tools they need to embed computational thinking and coding into their classrooms” (Saskatoon Industry Education Council, 2024a, para. 3). It does so by providing appropriate grade-level, hands-on activities, such as beginner block-based language coding for grades 6 to 9. Moreover, SaskCode emphasizes the need to integrate technology effectively in the classroom to help students build digital fluency, which is defined as “the ability to use digital technologies readily and strategically to learn, to work and to play” (Saskatoon Industry Education Council, 2024b, p. 2). In addition to SaskCode, there are other coding and computational thinking related resources available on the Government of Saskatchewan Curriculum website. For example, “Opportunities to Address Computational Thinking in the Grades 1 to 6 Mathematics and Science Curricula” is a free resource which outlines ways that teachers can integrate computational thinking in their classroom, including how it relates to curriculum expectations in that subject/grade (Saskatchewan Ministry of Education, 2021).

### **Manitoba**

Similar to Saskatchewan, there has been no updates to Manitoba’s curriculum to include coding or computational thinking in recent years, but there is and has been an emphasis on the inclusion of technology in the mathematics curriculum. For instances, one of the mathematical



processes included from kindergarten to grade 12 is technology visualization (Government of Manitoba, n.d.). Moreover, in the 2018 report, it was discussed that:

Several boards throughout the province ... are participating in Coding Quest in collaboration with The Learning Partnership, a program that teaches coding and video game development to students in grades 4-6 over a period of 25 sessions (The Learning Partnership, n.d.). The CT-focused program also utilizes a cross-curricular approach by incorporating various subjects, including science and mathematics. Though there are no present ministry initiatives in place to include CT as part of the provincial curriculum, Manitoba's participation in Coding Quest illustrates some focus on teaching coding skills to the province's students (Gannon & Buteau, 2018, p. 3).

There have been no new updates on the utilization of Coding Quest or any future plans to integrate coding and/or computational thinking more directly into the Manitoba curriculum.

### Ontario

At the time of the 2018 Gannon & Buteau report, there was no mandatory, province-wide, inclusion of coding or computational thinking in Ontario, but there was the start of many initiatives aiming to change that. It is safe to say that these initiatives were successful because since 2020/2021, coding and computational thinking have been integrated into the grades 1 to 9 mathematics curriculum (Broley et al., 2023). Broley et al.'s chapter summarizes how the use of coding is integrated into the algebra strand- "[t]he overall expectation is that students in Grades 1 to 8 ... will learn to solve problems and create computational representations of mathematical situations using coding concepts and skills" (Broley et al., 2023, p. 11). These skills and coding concepts are learned in a progression, beginning with sequential events, then nested events, followed by sub-programs and other control structures, and lastly, by the end of grade 9 "students are expected to be able to apply coding skills to represent mathematical concepts and relationships dynamically, and to solve problems, in algebra and across the other strands (e.g., number, data, spatial sense, and financial literacy)" (Broley et al., 2023, p. 11). Moreover, computational thinking "is presented as part of the transferable skills that 'are in high demand in today's globally connected world, with its unprecedented advancements in technology'" (Broley et al., 2023, p. 11).

In addition to the revised mathematics curriculum, the science curriculum was also updated in 2022 to include coding and computational thinking in grades 1 to 8 (Government of Ontario, 2022). Similar to computational thinking being a transferable skill in mathematics,

Skills developed through STEM education include computational thinking, coding, innovation, and scientific and engineering design. These skills are in high demand in today's globally connected world, as advancements in science and technology continue to impact all areas of our lives, and they form a critical component of the science and technology curriculum. Students use an engineering design process and associated skills to design, build, and test devices, models, structures, and systems, and they write and execute code in investigations and when modelling concepts (Government of Ontario, 2022, The Importance of STEM Education, para. 3).

Coding is mainly integrated in Strand A – STEM Skills and Connections where students evaluate the impact of technology and coding on everyday life and in STEM fields, as well as use coding in their investigations to model science concepts (Government of Ontario, 2022).



## Québec

Since the 2018 Gannon and Buteau report, there has been no changes to the Québec curriculum to integrate coding or computational thinking compulsorily. Thus,

Presently, Quebec has no formal integration of CT or coding within its curriculum. The Quebec Education Plan for mathematics, science, and technology suggests ways in which computers can be utilized in elementary mathematics and science, where the use of ICT is required, but how this is accomplished is at the teacher's discretion (Québec Ministère de l'Éducation et de l'Enseignement Supérieur, 2001). Listed suggestions have a basis in CT, including activities such as "learning to do a computer simulation of a random experiment" (Québec Ministère de l'Éducation et de l'Enseignement Supérieur, 2001, p. 21). Quebec's English school system offers online coding lessons, called Kids Code Jeunesse, for students between the ages of 7-12 (Learn Quebec, n.d.). The courses offered as part of this program emphasize CT through the teaching of programming languages and website building (Gannon & Buteau, p. 4).

However, there has been some local initiatives discussed in the "plan d'action numérique" (digital action plan) which aims to integrate programming in the classroom. For instance, at Alexander-Wolff primary school, students in grade 6 spend about an hour a week learning/exploring computer programming (Gouvernement du Québec, 2018). Moreover, "The action plan's 33 measures are intended to give new impetus to the shift to digital in the education system and to contribute actively to the development of Quebecers' digital skills" (Gouvernement du Québec, 2018, p. 9). In the second measure specifically, it is stated that "The Ministère will encourage and support the use of coding for educational and didactic purposes in order to help students acquire the learning and competencies prescribed in the Québec Education Program (QEP)" (Gouvernement du Québec, 2018, p. 27). According to the Action Plan, the use of coding in the classroom was hoped to be in most elementary and private schools by the 2020-2021 school year, but the "integration of coding will be driven by schools" (Gouvernement du Québec, 2018, p. 27). Thus, although coding is not officially required in Québec's curriculum, there are many local initiatives/ projects that have been and continue to be implemented at the school level.

## Newfoundland and Labrador

In earlier years, Newfoundland and Labrador's integration of coding and computational thinking has been limited; one of the only mentions of coding their curriculum appears in the 2016 grade 2 Science Curriculum. This curriculum introduces the idea of kinaesthetic coding, such as creating a step pattern of moving up, down, left, or right, to solve a problem (i.e., getting a car from point A to point B). Moreover, it is stated that as an extension students could be introduced to programmable robotics, computer programs or mobile device applications as a way to expose them to technology-based coding (Department of Education for Newfoundland and Labrador, 2016). However, there were no other cross-curricular mentions of coding until 2018. Starting in 2018, the grade 6 science curriculum "What Causes Eclipses?" unit was updated to now include the minor requirement to "Use coding (e.g., Scratch programming) to create a digital model of solar and lunar eclipses" (Department of Education for Newfoundland and Labrador, 2018, p. 87). It has since been a steady incline of coding integration into the curriculum. In 2021, a new Technology Education course, "Computer Science", was added into the grade 8 curriculum (Newfoundland and Labrador Education, 2023). This course begins to introduce students to the fundamental concepts of programming, including the planning,



designing, and creation of block-based programs for a given task (Department of Education for Newfoundland and Labrador, 2021). Then, in 2022, the kindergarten curriculum was updated to include more mentions of coding in other school subjects, such as science, stating “students may play with coding-related toys and technologies that develop their problem-solving abilities.” (Department of Education for Newfoundland and Labrador, 2022, p. 31) and “students may Identify problems to be solved when using coding-related toys and technologies” (p. 33).

### **Prince Edward Island**

In 2015, a news article was released pertaining to the discussion of integrating computer coding into Prince Edwards Island’s curriculum. The main advocate, Maureen Kerr, wanted to follow in Nova Scotia’s footsteps and make computer coding a required course of the elementary curriculum (Russell, 2015). However, there were no further updates on the status of this integration since, and as seen in the current curriculum, computer coding is still not a required course. Although, in 2018, CBC news published an article on P.E.I.’s participation in the international coding program Hour of Code, this appears to be the extent of coding that has been introduced in the P.E.I. classrooms.

### **Nova Scotia**

Nova Scotia is one of the provinces with the earliest integration of coding and computational thinking in its curriculum. More details can be found in the 2018 report, but in short,

Coding was first announced as a priority in 2015 for the province’s Education Action Plan... The Education Action Plan included a three-year goal, from 2016-2019, where all students would be introduced to coding, technology, and design (Province of Nova Scotia, 2015b). With the implementation of the ICT/Coding 4-6 Integration beginning August 2016, coding became mandatory for grade 4-6 students as one of eight of the program’s desired outcomes (Province of Nova Scotia, 2016b). The new curriculum for coding emphasizes the basics of computer science, including computational thinking and the use of algorithms, the sequencing of steps in a program, and debugging code (Province of Nova Scotia, 2016b). Writing programs to model real-world situations is also a component of programming education throughout these elementary grades (Province of Nova Scotia, 2016b). ... Beyond elective computer science courses offered in secondary school, Nova Scotia’s Ministry of Education and Early Childhood Development website provides a variety of resources to introduce and develop coding skills in grade 6-12 students (Nova Scotia Education and Early Childhood Development, n.d.) (Gannon & Buteau, p. 5).

Additionally, as part of Nova Scotia’s renewed grade 7 and 8 curriculum, Technology Education has been fully implemented province-wide (Nova Scotia Department of Education and Early Childhood Development, 2022). By the end of this course, the goal is that learner will be able to use a variety of “technological tools, processes, and applications; integrate technology education with other academic disciplines; design and create devices and objects that solve technological problems; and explain the consequences of technology and how it affects society” (Nova Scotia Department of Education and Early Childhood Development, 2022, para. 1).

### **New Brunswick**

This report, like the 2018 Gannon & Buteau report, focuses solely on New Brunswick's Anglophone curriculum. As discussed in the 2018 report, New Brunswick is another province





that has dedicated time to integrating coding and computational thinking into their curriculum (Gannon & Buteau, 2018). Since 2016, the Middle School Technology Education (MSTE) curriculum has been implemented, which

... has made coding and programming mandatory for students in grades 6 to 8 as part of its conceptual framework on Digital Technology Skills Exposure (New Brunswick Department of Education and Early Childhood Development, 2016). These skills are taught as components of the MSTE's general curriculum outcome (GCO) focusing on critical thinking and problem solving, highlighted in specific curriculum outcome (SCO) 2.5 that students in these grades should “understand and demonstrate computer coding/programming concepts and terminology” (New Brunswick Department of Education and Early Childhood Development, 2016, p. 19) (Gannon & Buteau, 2018, p. 5).

Moreover, it is required that a minimum of 10% of MSTE classes (per year) must be dedicated to coding. There have been no changes or updates to this curriculum since its implementation (Government of New Brunswick, 2024).

### Conclusion

Overall, there has been an increase in the (compulsory) integration of coding and computational thinking into the curriculum across the different provinces/territories. As of the current year (2024), nine provinces/territories have renewed programs integrating coding and/or computational thinking. These nine provinces/territories are comprised of; British Columbia, Nova Scotia, and New Brunswick, which had renewed programs as of the 2018 (Gannon & Buteau) report; as well as Alberta and Ontario which had programs still under discussion with the Ministry of Education/ had policies at the regional level as of 2018; and Newfoundland and Labrador which had very little integration in the 2018 report but has since added more; in addition to Yukon Territory, Northwest Territories, and Nunavut which were not previously explored in the 2018 report, but have been found to have programs with integrated coding/computational thinking. Québec and Manitoba still remain with mentions of future integration, but with no further updates at this time; and Prince Edward Island and Saskatchewan are still without mention of possible compulsory integration into their curriculum. Nevertheless, we are seeing an upward trend of integrating coding, computational thinking, or simply the use of technology in the classroom. Additionally, there are more and more available online and in-person resources for teachers or students to explore coding, such as the CanCode 3.0 Projects which includes Canada Learning Code, The Learning Partnership, and more (Government of Canada, 2023).

### Acknowledgments

I wish to thank Erica Huang (British Columbia), and Marie-Frédéric St-Cyr (Québec), for their feedback on the portions of the report related to the implementation of coding and computational thinking in their respective provinces.

### References

#### British Columbia

British Columbia Ministry of Education. (2018). *Applied Design, skills and technologies*.

Retrieved from <https://curriculum.gov.bc.ca/curriculum/adst>

British Columbia Ministry of Education. (2016). Area of learning: Mathematics kindergarten - big ideas. Retrieved from



[https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/enmathematics\\_k-9\\_elab.pdf](https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/enmathematics_k-9_elab.pdf)

British Columbia Ministry of Education. (n.d.). *Introduction to Applied Design, Skills, and Technologies*. Retrieved from <https://curriculum.gov.bc.ca/curriculum/adst/introduction>

British Columbia News. (2016, September 6). *Ten things to know about B.C.'s new curriculum*. British Columbia Government News Releases. Retrieved from <https://news.gov.bc.ca/releases/2016EDUC0082-001558>

### **Yukon Territory**

Government of Yukon. (2023, July 6). *Learn about the Yukon's School Curriculum*. Yukon's curriculum | Home. Retrieved from <https://yukon.ca/en/school-curriculum>

### **Alberta**

Alberta Education. (2017). DRAFT kindergarten to grade 12 mathematics scope and sequence. Retrieved from [https://cdnsm5-ss7.sharpschool.com/UserFiles/Servers/Server\\_12737792/File/Departments/Curriculum/New%20Curriculum%20Development/Math%20K-12%20Scope%20and%20Sequence.pdf](https://cdnsm5-ss7.sharpschool.com/UserFiles/Servers/Server_12737792/File/Departments/Curriculum/New%20Curriculum%20Development/Math%20K-12%20Scope%20and%20Sequence.pdf)

new LearnAlberta. (2023, March). Science Kindergarten to Grade 6 Curriculum. Retrieved from <https://curriculum.learnalberta.ca/curriculum/en/c/sci5?s=SCI>

### **Northwest Territories**

Government of the Northwest Territories. (n.d.-a). *Mathematics*. Education, Culture and Employment. Retrieved from <https://www.ece.gov.nt.ca/en/services/jk-12-school-curriculum/mathematics>

Government of the Northwest Territories. (n.d.-b). *Science*. Education, Culture and Employment. Retrieved from <https://www.ece.gov.nt.ca/en/services/jk-12-school-curriculum/science>

Government of the Northwest Territories. (2004). NWT K-6 Science and Technology Curriculum (2004). Retrieved from [https://www.ece.gov.nt.ca/sites/ece/files/resources/k-6\\_science\\_technology\\_curriculum.pdf](https://www.ece.gov.nt.ca/sites/ece/files/resources/k-6_science_technology_curriculum.pdf)

Lachacz, A. (2023, February 6). *N.W.T to complete curriculum transition by 2028, a “sad commentary” for Alberta*. CBC News Edmonton. Retrieved from <https://edmonton.ctvnews.ca/n-w-t-to-complete-curriculum-transition-by-2028-a-sad-commentary-for-alberta-1.6262179#:~:text=After%20using%20Alberta's%20education%20curriculum,by%20207%2D28%2C%20the%20N.W.T.>

### **Nunavut**

Department of Education Ilinniaqtuliyikkut Ministère de l'Éducation. (2019, August). 2019 – 2020 Nunavut Approved Curriculum and Teaching Resources. Retrieved from [https://www.gov.nu.ca/sites/default/files/documents/2022-03/2019-20\\_nunavut\\_approved\\_curriculum\\_and\\_teaching\\_resources.pdf](https://www.gov.nu.ca/sites/default/files/documents/2022-03/2019-20_nunavut_approved_curriculum_and_teaching_resources.pdf)

Department of Education Ilinniaqtuliyikkut Ministère de l'Éducation. (n.d.). Curriculum and Teaching Resources Orientation 2023/2024. Retrieved from [https://www.gov.nu.ca/sites/default/files/documents/2023-12/CurriculumAndTeachingResources\\_2023-2024.pdf](https://www.gov.nu.ca/sites/default/files/documents/2023-12/CurriculumAndTeachingResources_2023-2024.pdf)

### **Saskatchewan**



Saskatchewan Ministry of Education. (2021). *Curriculum Connections - Opportunities to Address Computational Thinking in the Grades 1 to 6 Mathematics and Science Curricula*. Saskatchewan Curriculum – Resources. Retrieved from [https://curriculum.gov.sk.ca/DisplayResource?resource\\_id=63190&curric\\_id=-1&language=en](https://curriculum.gov.sk.ca/DisplayResource?resource_id=63190&curric_id=-1&language=en)

Saskatoon Industry Education Council. (2024a). *Robotics For K-12*. SaskCode. Retrieved from <https://saskcode.ca/>

Saskatoon Industry Education Council. (2024b). *SASKATCHEWAN K–12 COMPUTATIONAL THINKING AND CODING FRAMEWORK*. SIEC\_SASKCODE\_CTC\_FRAMEWORK.PDF. Retrieved from [https://drive.google.com/file/d/1uLu65\\_ngrPDZ3kayPdsXHdtZkUususvyx/view](https://drive.google.com/file/d/1uLu65_ngrPDZ3kayPdsXHdtZkUususvyx/view)

## Manitoba

Government of Manitoba. (n.d.). *Education and early childhood learning*. Mathematics | Manitoba Education and Early Childhood Learning. Retrieved from <https://www.edu.gov.mb.ca/k12/cur/math/overview.html>

The Learning Partnership. (n.d.). Coding quest. Retrieved from <https://www.thelearningpartnership.ca/what-we-do/student-programs/coding-quest>

## Ontario

Broley, L., Buteau, C., Modeste, S., Rafalska, M., & Stephens, M. (2023). Computational Thinking and Mathematics. In *Handbook of Digital Resources in Mathematics Education* (pp. 1–38). Springer International Publishing. Retrieved from <https://hal.science/hal-04536404/document>

Government of Ontario. (2022). *Science and Technology (2022)*. Ontario Curriculum and Resources. Retrieved from <https://www.dcp.edu.gov.on.ca/en/curriculum/science-technology/context/fundamental-concepts>

## Québec

Gouvernement du Québec. (2018). DIGITAL ACTION PLAN FOR EDUCATION AND HIGHER EDUCATION. Retrieved from [https://www.education.gouv.qc.ca/fileadmin/site\\_web/documents/ministere/PAN\\_Plan\\_action\\_VA.pdf](https://www.education.gouv.qc.ca/fileadmin/site_web/documents/ministere/PAN_Plan_action_VA.pdf)

Learn Quebec. (n.d.). Online coding classes. Retrieved from <http://www.learnquebec.ca/kids-codejeunesse>

Québec Ministère de l'Éducation et de l'Enseignement Supérieur. (2001). Chapter 6: mathematics, science, and technology. Retrieved from [http://www.education.gouv.qc.ca/fileadmin/site\\_web/documents/PFEQ/educprg2001-061.pdf](http://www.education.gouv.qc.ca/fileadmin/site_web/documents/PFEQ/educprg2001-061.pdf)

## Newfoundland and Labrador

Department of Education for Newfoundland and Labrador. (2016). *Science 2 curriculum guide 2016*. Retrieved from [https://www.gov.nl.ca/education/files/k12\\_curriculum\\_guides\\_science\\_science\\_2\\_2016\\_fi nal.pdf](https://www.gov.nl.ca/education/files/k12_curriculum_guides_science_science_2_2016_fi nal.pdf)

Department of Education for Newfoundland and Labrador. (2018). *Science 6 Curriculum Guide 2018*. Retrieved from [https://www.gov.nl.ca/education/files/Science\\_6\\_2018.pdf](https://www.gov.nl.ca/education/files/Science_6_2018.pdf)



Department of Education for Newfoundland and Labrador. (2021). *Computer science 8 Curriculum Guide 2021*. Retrieved from

[https://www.gov.nl.ca/education/files/Computer\\_Science\\_8\\_8-13-21.pdf](https://www.gov.nl.ca/education/files/Computer_Science_8_8-13-21.pdf)

Department of Education for Newfoundland and Labrador. (2022). *Science Kindergarten Curriculum Guide 2022*. Retrieved from

[https://www.gov.nl.ca/education/files/science\\_kindergarten\\_2022.pdf](https://www.gov.nl.ca/education/files/science_kindergarten_2022.pdf)

Newfoundland and Labrador Education. (2023, October 18). *Technology education*. Retrieved from <https://www.gov.nl.ca/education/k12/curriculum/guides/teched/>

### **Prince Edward Island**

CBC News. (2018, December 6). *P.E.I. students program robots as part of Global hour of code | CBC News*. CBCnews. Retrieved from <https://www.cbc.ca/news/canada/prince-edward-island/pei-students-coding-hour-of-code-1.4933271>

Russell, N. (2015, November 12). *Kids need to be coding as soon as they start school, says Tech Advocate | CBC news*. CBCnews. Retrieved from <https://www.cbc.ca/news/canada/prince-edward-island/pei-computer-coding-students-1.3314708>

### **Nova Scotia**

Nova Scotia Department of Education and Early Childhood Development. (2022). *Technology education 7: Education & early childhood development*. Technology Education 7 | Education & Early Childhood Development. Retrieved from

<https://curriculum.novascotia.ca/english-programs/course/technology-education-7>

Nova Scotia Education and Early Childhood Development. (n.d.). Hour of code. Retrieved from <https://medialibrary.ednet.ns.ca/hour-code>

Province of Nova Scotia. (2015b). The 3Rs: Renew, refocus, rebuild – Nova Scotia’s action plan for education. Retrieved from

<https://www.ednet.ns.ca/sites/default/files/docs/educationactionplan2015en.pdf>

Province of Nova Scotia. (2016b). Information and communication technology/coding 4-6 integration. Retrieved from [https://www.ednet.ns.ca/files/curriculum/infotech\\_coding\\_4-6\\_streamlined.pdf](https://www.ednet.ns.ca/files/curriculum/infotech_coding_4-6_streamlined.pdf)

### **New Brunswick**

Government of New Brunswick. (2024). *Curriculum development (anglophone sector)*. Government of New Brunswick, Canada. Retrieved from

[https://www2.gnb.ca/content/gnb/en/departments/education/k12/content/anglophone\\_sector/curriculum\\_anglophone.html#2](https://www2.gnb.ca/content/gnb/en/departments/education/k12/content/anglophone_sector/curriculum_anglophone.html#2)

New Brunswick Department of Education and Early Childhood Development. (2016). Middle school technology education. Retrieved from

<https://www2.gnb.ca/content/dam/gnb/Departments/ed/pdf/K12/curric/TechnologyVocational/Middle%20School%20Technology.pdf>

### **Other**

Bocconi, S., Chiocciariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). Developing computational thinking in compulsory education: Implications for policy and practice. European Union. Retrieved from

<https://publications.jrc.ec.europa.eu/repository/handle/JRC104188>



- Bocconi, S., Chiocciariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M.A., Jasutė, E., Malagoli, C., Masiulionytė-Dagienė, V. and Stupurienė, G. (2022). Reviewing Computational Thinking in Compulsory Education. Inamorato Dos Santos, A., Cachia, R., Giannoutsou, N. and Punie, Y. editor(s), Publications Office of the European Union, Luxembourg, 2022, ISBN 978-92-76-47208-7, doi:10.2760/126955, JRC128347. Retrieved from <https://publications.jrc.ec.europa.eu/repository/handle/JRC128347>
- Gannon, S., & Buteau, C. (2018). Integration of Computational Thinking in Canadian Provinces. (2018). In *Online Proceedings of the Computational Thinking in Mathematics Education Symposium*, UOIT (Scarborough), October 2017. Retrieved from [http://ctmath.ca/wp-content/uploads/2018/10/Symposium\\_CanadaMap\\_Gannon-Buteau.pdf](http://ctmath.ca/wp-content/uploads/2018/10/Symposium_CanadaMap_Gannon-Buteau.pdf)
- Government of Canada. (2023, June 23). *CanCode 3.0 Project Descriptions*. Minister of Innovation Science and Economic Development. <https://ised-isde.canada.ca/site/cancode/en/funded-cancode-initiatives>
- Stephens, M., Kadjevich, D., M., Qinqiong, Z. 14th International Congress on Mathematical Education. (2021). Retrieved from <https://www.mi.sanu.ac.rs/~djkadij/ICME14DG1.pdf>





## Computational Thinking: A Reflection on the Symposium by an Early Career Researcher in Undergraduate Mathematics Education

Francis Duah  
Toronto Metropolitan University

### Something(s) that Struck Me

I have attended three major in-person events on learning and teaching in undergraduate mathematics since summer 2022. While the sample size of the events is small, for me, the symposium on *Coding, Computational Modelling, and Equity in Mathematics Education* tops the list. What a week it was! What an organizing team!

The organizing committee put on a great academic event and made all participants feel welcome. The symposium offered me a personal and professional opportunity to learn and reflect on a new research area - *Computational Thinking*.

One thing that struck me is the fact that computational thinking transcends K-12, post-secondary, and undergraduate mathematics education. Yet, evidence from my classroom practice suggests that coding is not necessarily an enjoyable activity for most undergraduate mathematics students. I therefore wonder what we could do to motivate lower level students to see the need for them to develop coding competency and computational thinking skills. This calls for some further research work on task selection and task design.

I enjoyed the event so much that I did not want it to end. I wanted more time at Brock to learn about MICA, and shadow one lower level as well as one upper level undergraduate mathematics student; and hear more about some of the work the Brock team have done including their contributions to *Educational Studies in Mathematics*. Nonetheless, I returned to my affiliated institution inspired and re-invigorated to carve a research path that I hope will widen participation in undergraduate mathematics education.

### Something(s) I Learned or Gained

I had not read any paper on *Computational Thinking* before the symposium, and I did not have the knowledge base to engage in any intellectual debate on the topic. So, to get up to speed with the subject, the [University of York](https://online.york.ac.uk/what-is-computational-thinking/), not [York University](#), came to my rescue at <https://online.york.ac.uk/what-is-computational-thinking/>. This article describes computational thinking in a way that will entice anyone new to the subject to engage with it. After attending the symposium and engaging with the professional development day and working group activities, I have also come to believe that undergraduate mathematics faculty need to emphasize in their teaching the “four Cs” of 21st century learning – communication, critical thinking, collaboration, and creativity (Grover, 2018) – and coding in relevant mathematics courses is a place to start. As an academic tribe, the undergraduate mathematics faculty of which I am part, still have a challenge and a long way to go to ensure that their students are fully prepared for new and emerging careers that draw on computational thinking. The reason for this is because much of our practice – teaching and assessment – privileges tests and exams, correct answers, procedural knowledge, and routine tasks over creativity, debate, and open-ended tasks.

At the professional development day, I gained working knowledge of coding at the Callysto Hub platform. Since then, I have introduced my undergraduate students to the platform as an alternative to R Studio. I see an opportunity for an outreach programme at Toronto Metropolitan University for high school students within the GTA area.



### **Something(s) I Now Wonder**

I have always been interested in quantitative social science, computational social science, and computational statistics. We know that Data Science has become an emerging field with new careers on the horizon. I wonder if we could offer a mathematics course that focuses on addressing social problems through the application of mathematics and/or statistics. I think such a course will require computational thinking skills, and yet it will be unique and distinct from data science.

### **References**

- Brooks, R. (2023). What is Computational Thinking? University of York: Available at <https://online.york.ac.uk/what-is-computational-thinking/>. [Accessed on 2023-11-23].
- Callysto (2023) Callysto Hub. Available at <https://hub.callysto.ca/>. [Accessed on 2023-11-23].
- Grover, S. (2018). The 5th 'C' of 21st Century Skills? Try Computational Thinking (Not Coding), Available at <https://www.edsurge.com/news/2018-02-25-the-5th-c-of-21st-century-skills-try-computational-thinking-not-coding>. [Accessed on 2023-11-23].



## A Perspective of the Experience at CCMEME Symposium

Carolina Yumi Lemos Ferreira Graciolli

São Paulo State University, Brazil, [carolina.graciolli@unesp.br](mailto:carolina.graciolli@unesp.br)

I am a PhD student in Mathematics Education at São Paulo State University, Brazil. My research is about origami and Mathematics Education, mainly addressing questions about how paper folding can contribute to the production of mathematical knowledge. Paper folding and mathematics education have been studied to support learning environments (Yazlik & Çetin, 2023), to assist in the production of geometric knowledge (Natalija, Lavicza, Fenyvesi & Milinkovic, 2020), also to explore spatial visualization and development of reasoning (Arici & Aslan-Tutak, 2015) or for the development of aspects of computational thinking (Graciolli, Rocha Junior & Scucuglia, 2022; Budinski, Lavicza, Fenyvesi & Novta, 2019).

The interest in participating in the “Coding, Computational Modelling, & Equity in Mathematics Education Symposium” arose after the invitation from my advisor, Professor Dr. Ricardo Scucuglia, to contribute to the production and presentation of a poster about some research that was developed in Brazil about computational thinking. Furthermore, during my time as Visiting Research Only (VRO) at Western University, under the guidance of Professor Dr. Immaculate Kizito Namukasa, she also told me about the event during an orientation meeting.

During the event I had the opportunity to present, together with my advisor, our work entitled “Research on Aspects of Computational Thinking in Brazil”. All the comments and questions were very interesting and made me think about other aspects that can be developed in the research. In addition to presenting, I was able to learn about the research of other participants at the event, I learned about different approaches and software, such as Virtual Reality, Augmented Reality, block programming, unplugged activities, different languages and Scratch, CoSpace, ThinkerCad, Micro:Bit, Ozobots, GeoGebra, ChatGPT, among others.

I learned about different perspectives on computational thinking and computational Modelling (Gadanidis, 2017; Brennan & Resnick, 2012; Gadanidis, Hughes, Namukasa & Scucuglia, 2019) and the importance of discussing such issues in relation to Mathematics Education. The lectures were important for opening possibilities beyond the perspectives already studied in Brazil. I was surprised by the different possibilities of articulating computational thinking with art, algebra, mathematical thinking, justice, equity, inclusion, racism, among others. Furthermore, it was very interesting to listen to researchers from different parts of the world, both in the lectures and in the poster sessions, I had contact with people from Canada, Netherlands, Brazil, United Kingdom, United States of America, Colombia, France, Australia, South Korea, Mexico, etc.

In conclusion, with the event I was able to broaden my research and practice horizons. Mainly during the discussion group meetings “Equity, diversity, and inclusivity in coding and computational Modelling in mathematics education” where I had the opportunity to think about situations in which coding could be part of an action towards equity, diversity, and inclusivity. The discussions raised in this group made me rethink origami as an opening for computational thinking (Graciolli, Rocha Junior & Scucuglia, 2022) and consider the possibilities of investigating origami, fractals, and coding. Something I wonder now is how activities with paper folding and coding can contribute to mathematical education and what we should consider for equity, diversity, and inclusivity.



## References

- Arici, S., & Aslan-Tutak, F. (2015). The effect of origami-based instruction on spatial visualization, geometry achievement, and geometric reasoning. *International Journal of Science and Mathematics Education*, 13, 179-200.
- Brennan, K.; Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *AERA 2012*, Vancouver, Canada.
- Budinski, N., Lavicza, Z., Fenyvesi, K., & Novta, M. (2019). Mathematical and Coding Lessons Based on Creative Origami Activities. *Open Education Studies*, 1(1), 220-227.
- Budinski, N., Lavicza, Z., Fenyvesi, K., & Milinković, D. (2020). Developing primary school students' formal geometric definitions knowledge by connecting origami and technology. *International Electronic Journal of Mathematics Education*, 15(2).
- Gadanidis, G. (2017). Five Affordances of Computational Thinking to Support Elementary Mathematics Education. *Journal of Computers in Mathematics and Science Teaching*, v.36 n.2, 143-151.
- Gadanidis, G., Hughes, J. M., Namukasa, I., & Scucuglia, R. (2019). Computational modelling in elementary mathematics teacher education. In: *International Handbook of Mathematics Teacher Education: Volume 2* (pp. 197-222).
- Graciolli, C. Y. L. F, Rocha Junior, R., & Scucuglia, R. R. da S. (2022). Aspects of computational thinking in unplugged activities with origami and mathematics. *Dialogia*, 40, 1-20, e21513.
- Yazlik, D. Ö., & Çetin, İ. (2023). Creating Learning Environments with Origami: Experiences of Pre-Service Mathematics Teachers. *Journal of Pedagogical Research*, 7(3), 174-193.



## A Reflection on the Coding, Computational Modelling, and Equity in Mathematics Education Symposium

Sheree Rodney  
Ontario Tech University

Computational thinking (CT) and mathematical thinking (MT) are two significant skills that are required to better understand how to leverage technology and solve problems in the 21<sup>st</sup> century. The recently concluded Coding, Computational Modelling, and Equity in Mathematics Education (CCMEME) symposium (a FIELDS-sponsored event) at Brock University, was enlightening and acts as a reminder that technology continues to play an important role in mathematics education. It was refreshing to reconnect with mathematics education researchers that I have not seen since the pandemic. The event was well organized and focused on a deeper understanding of CT and its relationship to MT. In addition, the symposium addressed ways in which computational Modelling can be applicable to mathematics teaching and learning, as well as, addressing issues pertaining to equity. That is, how to promote equity in participation when learners engage in technology-enhanced learning? My interest in the symposium arose from my dissertation work on technology integration in mathematics education and most recently, my interest in Equity, Diversity, and Inclusion (EDI) principles. I saw the CCMEME as an opportunity to expand my knowledge in both areas.

The keynote speakers' addresses were thought-provoking and presented ideas around understanding the knowledge required to effectively implement technology in education. Furthermore, the discussion about cultural issues and ways in which AI technology might propagate racial bias, highlighted the need for an awareness not solely for research or education purposes, but also how these issues relate to our personal experiences.

I attended the *Equity, Diversity, and Inclusivity working group* because I needed a deeper understanding of how computational and mathematical thinking might contribute to equity, diversity, and inclusion, and how this understanding might shape my classroom practice. The aim of this working group was to address issues of quality equitable pedagogy and the benefits of coding to mathematics education. To accomplish this goal, we were engaged in rich discussions and explorations of existing activities to identify how they might be adapted to promote equitable experiences for all learners. One of the things I have struggled with in the past, was being able to identify resources that focus on CT and MT as a unified whole rather than separate entities.

As a result, I was impressed by the variety of activities requiring these two skills that were shared by both participants and facilitators during the sessions.

Attending the symposium highlighted the importance of understanding how these distinctly different, yet similar skills are closely connected to each other. I learned that although CT draws on principles and techniques relating to computer science, it does involve similar elements to MT; such as pattern recognition and problem-solving. I also learned that several jurisdictions are now embracing computational and mathematical thinking for interdisciplinary collaboration and innovative solutions, particularly in mathematics education, and that there is an increased interest in research about the intersection of these two related concepts.

The working group sessions piqued my curiosity, in particular, on the implications of computational Modelling to mathematics teaching and learning. As a result, I would like to pursue the following ideas further:





Rodney, S. (2023). A Reflection on the Coding, Computational Modelling, and Equity in Mathematics Education Symposium. In Online Proceedings of the *Coding, Computational Modelling, and Equity in Mathematics Education Symposium*, St. Catharines (Canada), April 2023.

1. How can I take into consideration identities such as race, gender, religion, socioeconomic status, disability, and marginalized groups in the context of CT and MT to address tensions that learners might face?
2. How can CT and MT bridge the gap between technology and societal needs?



## Some Reflections Emerged from the Conference

Marie-Frédéric St-Cyr  
UQAM

I registered for the conference with enthusiasm, as I was finally going to be able to meet the researchers behind the papers I have been reading for a little over three years. I was finally going to be able to meet the humans behind the ideas that have shaped my master's thesis and are now influencing my doctoral project. Would these people be willing to discuss with me? Would listening to them clarify the ideas I have read? I had no idea, but I was eager to exchange and learn.

After the conference, I better understood my position in relation to algorithmic thinking in mathematics. Indeed, throughout the presentations and exchanges, I could see that some perspectives were closer to mine and others more distant. I became aware that my perspective was that of a mathematics educator. I can engage in discussions with people who are specifically interested in computational or algorithmic thinking in different contexts, but my viewpoint remains that of a mathematics educator. However, I understood the importance of developing my programming skills. Indeed, through discussions and explorations in my working group, I noted that proficiency in programming provides freedom and space to create or explore diverse and original tasks. Since then, I have been practicing programming to achieve this freedom and to explore ideas that can guide my theoretical and practical reflections.

Furthermore, Richard Noss's conference guided my research intentions for my thesis. He emphasized the importance of creating tasks for students and contributing to the training of teachers related to the development of computational thinking in the classroom. In the context of my thesis, I aim to contribute to these reflections by proposing collaboration with teachers to integrate algorithmic thinking into their mathematics classes, creating tasks for them to experiment with. These experiments, along with reflections from working with teachers, will allow me to continue theoretical and epistemological reflections related to algorithmic thinking in mathematics.

Finally, during the conference, I met extraordinary people with whom I exchanged ideas, leading me to connect with individuals interested in computational thinking in Montreal. Consequently, I expanded my network and opened up possibilities for research related to computational thinking, computer science, and algorithmic thinking. During the conference, I also had the opportunity to meet Chantal Buteau, who agreed to host me for an internship at Brock in the winter of 2024.

I want to express my gratitude to the organizing committee for making this event possible.