

Professional Development in Coding and Computational Thinking for Mathematics Teachers

Keng Cheng Ang

kengcheng.ang@nie.edu.sg

National Institute of Education
Singapore

Marc Yi Fei Yeo

marc.yeo@nie.edu.sg

National Institute of Education
Singapore

Abstract: *Since 2006, computational thinking (CT) has been popularised as a critical interdisciplinary skill and linked to mathematical thinking, solidifying its applicability in mathematics education. Singapore has actively introduced CT in its mathematics curriculum and provided professional development (PD) opportunities for mathematics teachers to develop their competencies in incorporating CT in mathematics classrooms. However, research examining how and whether such PD prepares educators to teach mathematics using CT is scarce. The study fills this gap by examining how a PD course that introduces the VBA coding language in Microsoft Excel for computational problem solving develops mathematics educators' coding skills and CT, and how participating in-service teachers perceive the course with regards to learning coding. Qualitative analysis of the course design revealed that the course materials are capable of helping learners develop CT through instilling in them certain coding habits, while qualitative and quantitative analysis of Likert-scale and open-ended responses in the course feedback highlighted many strengths and suggested areas for improvement in various aspects of the course, like course structure, course materials, level of course difficulty, and perceived usefulness and applicability of the course. These findings reveal the benefits of computational approaches adopted in this study for developing CT and coding skills, the relevance of such approaches in mathematics education, areas that can potentially be improved for more effective PD, as well as how rich insights generated by feedback and course design analysis can contribute to assessing the impact of PD and tailoring PD courses to specific teacher needs and concerns.*

1. Introduction

One of the earliest mentions of computational thinking (CT) was made by Papert [23, p. 17] in 1980 when he suggested that “computational technology and computational ideas” could create “new possibilities for learning, thinking, and growing”. This term gained prominence when Wing [31] re-introduced it in 2006 as an essential transdisciplinary skill central to problem solving. While there is no consensus on what constitutes CT, the literature generally recognises CT as a mental procedure of pinpointing problems and devising solutions that can be executed via computational means [18], foregrounding abstraction (extracting details relevant for problem solving), algorithmic thinking (approaching problems in steps), pattern recognition (identifying patterns in problems for understanding other problems), and decomposition (dividing problems into smaller, manageable parts) as core CT skills [10].

Numerous overlaps between CT and mathematical thinking suggest the relevance of computational approaches in mathematical problem solving. CT's computer science origins are founded on mathematical thinking [31], and scholars have recognised pedagogical benefits of CT in learning mathematical concepts and skills [30]. Frameworks linking CT to mathematics-specific skills include Sneider et al. [28] and Weintrop et al. [30], who respectively suggest areas where mathematical thinking and CT overlap as well as shared CT practices in mathematics and science.

There is also a close link between CT and mathematical modelling – both approaches share a common goal of problem solving as mathematical models are an “abstraction” of problems “in a mathematical form”, and furthermore, modelling and computational tools are associated with enhanced mathematical skills [1, p. 1, 16, 27]. Pertaining to computational tools, coding has been identified as highly effectual for cultivating CT [20]. However, knowledge alone of using computational tools is not identical to developing CT [3, 26] – while such tools can execute solutions, a crucial step lies in generating solutions using a computational approach, which Ang [2, p. 127] terms *habits of mind*, or “mental habits” of those who utilise computational “tools... constructs... methods... to solve problems”. He argues that coding creates opportunities to cultivate these CT habits, such as extracting relevant information in problems and transforming them into variables (i.e., abstraction).

Since Wing’s article, there has been a marked increase in the educational integration of CT. Singapore for example has actively promoted CT since 2014, focusing the curricular integration of CT on mathematics. CT and computational tools are at the heart of Singapore’s mathematics syllabus, with both featuring prominently in the secondary and junior college (i.e., Grades 7 to 12) levels. However, not many educators in Singapore are well acquainted with CT and how to use it to teach mathematics [7], with most having no prior exposure to coding in university or pre-service education [13]. Given the increasing pressure to prepare students for a workforce that favours computational proficiency, teachers need more opportunities to upgrade their CT capabilities.

This study examines the course design and feedback of a professional development (PD) course on computational approaches for mathematical problem solving. It is targeted at in-service mathematics teachers in Singapore and introduces the use of Visual Basic for Applications (VBA) in Excel as a tool for mathematical problem solving. It aims to answer the following research questions:

RQ1: How does the course cultivate mathematics educators’ coding skills and CT?

RQ2: How do participants perceive the course with regards to learning and/or coding?

Excel is chosen as the computational tool for this course because of the pervasive use of Excel spreadsheets, with the number of users of spreadsheets as a coding tool estimated to be 11 million in the U.S. alone despite many users having minimal or no coding knowledge [9]. Many teachers and students have ready access to Excel at little to no extra cost and are more familiar with it compared to specialised coding software, as Excel is available on most computers and resembles the pen and paper format students are accustomed to [25]. Teachers can therefore implement Excel applications in class as a “natural and logical extension” of their existing proficiencies [13, p. 82]. Excel is also versatile as it can solve most school-level mathematical problems, contains numerous features that students can experiment with independently, and potentially enables prolonged information retention when mathematics is taught through it [9, 25].

2. Developing CT competencies through coding

There are numerous studies involving PD courses for developing mathematics teachers’ capabilities in teaching using CT. These mainly examine program outcomes like changes in teachers’ CT knowledge, skills, and perceptions. Gadanidis et al. [12] examine shifts in attitudes towards and knowledge of CT amongst pre-service primary level teachers in Canada who attended a course on integrating CT in mathematics. Analyses of assignments and discussions showed that the teachers shifted from having little to no knowledge of CT to being able to identify its integration in

mathematics and other subjects and its societal relevance. They also reported decreased anxiety towards CT and a growth mindset towards learning and teaching CT. Using participant observation and interviews, Martins [21] examines how primary level Portuguese pre-service teachers attending a mathematics pedagogy course perceive the link between CT and mathematics education, and ways in which they create learning activities that bolster skills in both areas. The teachers initially expressed unfamiliarity with CT aspects such as whether CT and mathematical skills are equivalent. However, as they worked on a mathematical task aimed at cultivating CT competencies, they gained familiarity in how to implement such tasks and honed their understanding of CT.

With regards to studies on PD courses on coding, Humble and Mozellius [15] analyse Grade 7 to 12 Swedish mathematics and technology educators' perceptions of CT during a PD session on coding. Essays submitted describe opportunities and challenges that teachers attributed to each aspect of CT and the general application of CT in school. They perceived different aspects as useful for their pedagogy and students' learning, albeit noting that CT may not always be relevant to the subject at hand and could be challenging to implement for reasons such as the perceived time-consuming nature of CT integration. Araujo et al. [4] on the other hand, examine knowledge related to CT acquired by Grade 7 to 12 pre-service teachers in Canada during a teacher education course on CT in mathematics and science education which covers coding. Analysis of discussion and reflection posts showed that their experiences of acquiring CT comprise four themes: pedagogical approaches (e.g., incorporation of CT in subjects), affordances (e.g., potential of CT in enhancing mathematics education), tools (e.g., affordances of CT tools for learning mathematics), and definitions (e.g., gaining a deeper understanding of CT).

These studies suggest the effectiveness of PD in improving mathematics teachers' perceptions, knowledge, and integration of CT. However, despite close links between mathematics learning and coding [22], there is little research on (and availability of) PD that introduces mathematics educators to coding [19] and virtually no studies evaluating the use of Excel in PD for mathematics teachers. Given the lack of coding proficiency amongst mathematics educators [13], this should be addressed so that appropriate PD can be introduced. This study therefore is an attempt to address this urgent need.

Studies on pedagogy involving CT typically examine CT concepts/skills/practices developed, mathematics topics taught through the use of computational tools, and how these tools augment the understanding of mathematics and CT. Chan et al. [8] identify as many as 64 tools in mathematics, with coding languages being the most common, digital tangibles and formative/summative assessments in second place, followed by apps/games and other digital tools (e.g., spreadsheets). Top CT skills and mathematics topics assessed using these tools include algorithmic thinking, abstraction, debugging, loops, and sequences for skills, and geometry and measurement, number and operations, algebra, statistics and probability, and calculus for topics. Kong [17] on the other hand observes that primary school students in Hong Kong develop CT concepts like conditionals, CT practices like debugging, and CT perspectives like expressing knowledge of mathematical concepts, as they learn to code an app for determining composite and prime numbers.

Several studies examine how Excel develops CT and mathematical thinking. Ang [3] discusses how simulating mathematical models in Excel helps to develop CT as mental habits, which include step-by-step coding and making assumptions to simplify problems into a form that can be expressed mathematically for computation. Ho et al. [14] and Sanford [25] introduce coding instructions for mathematical problems that are solvable on Excel, while van Borkulo et al. [29] demonstrate how CT can be developed through coding lessons on statistics in Excel. Combining data like questionnaires, interviews, and spreadsheets, the latter analysed how coding in Excel enables students to develop CT in the areas of computational practices, perspectives, and concepts (see [6]).

These studies suggest that there are several CT tools which mathematics teachers can be exposed to, and many mathematical topics where these tools may be applied in. However, research on the general use of tools like Excel in mathematics instruction remains scarce. Given the advantages of Excel in mathematics teaching and learning, there is much value in research which provides concrete, targeted examples of how Excel can be harnessed to teach mathematical concepts while developing one's competencies and habits in CT.

3. The present study

3.1 Data collection and selection

This study analyses two datasets: the course design of a PD course on computational approaches for mathematical problem solving and the participants' post-course feedback. The latter comes from a standard feedback form, which consists of a 5-point Likert-scale and open-ended questions. The selected dataset contains 125 open-ended responses and 364 Likert-scale responses collected from 42 participants across three runs of the course.

3.2 Data analysis

A mixed methods analysis was conducted where the open-ended responses and course design were analysed qualitatively using thematic analysis, and the Likert-scale responses quantitatively using percentages. Thematic analysis involves "identifying, analysing and reporting patterns (themes) within data" [5, p. 79]. The course design was analysed first, by outlining the topics taught and linking instructional instances in the course materials to attempts to develop CT habits through coding. This answers RQ1 on how this course develops coding and CT skills.

The analysis of the feedback focuses on strengths and areas for improvement in the course as suggested by participating educators, as well as their perceptions of the usefulness and applicability of the course in relation to mathematics teaching and other contexts. The findings of this analysis are presented in tables and bar charts with a thematic breakdown.

The Likert-scale responses were assigned to the following categories: 1) instructor's teaching, 2) course learning outcomes, 3) course structure, 4) course materials, and 5) likeliness to recommend the course, while the open-ended responses were assigned to the following themes: 1) course delivery, 2) course difficulty, and 3) usefulness and applicability of course content. Themes for the open-ended responses are decomposed into subthemes whose associated feedback are further categorised into strengths and areas for improvement in the course, summarised, and presented in a table, while categories for the Likert-scale responses are presented with associated statements in a table and the percentages of participants' level of agreement with each statement are shown in a bar chart. The analysis of these responses answers RQ2 on discerning participants' views about the PD.

The next two sections present the analysis of the course design followed by a quantitative and qualitative examination of the feedback.

4. Design of the PD course

This section analyses the design of the PD course entitled "*Computational Approaches for Mathematical Problem Solving*", which was conducted by the first author over three runs in 2022 and 2024 and comprises six sessions per run. The course was open to mathematics teachers in Singapore's junior colleges (Grade 11 to 12 schools) who have a basic understanding of Excel. It

aims to raise their confidence in using easily accessible computational tools like Excel, help them realise the potential of such tools in learning mathematics, and develop their CT habits via exposure to Excel with VBA in mathematical problem solving.

The course is guided by the following learning objectives: 1) Learning fundamentals of VBA coding in Excel for mathematical problem solving; 2) Utilising Excel and VBA to perform mathematical computations and problem solving; 3) Designing computational solutions using concepts and tools taught in this course; 4) Gaining exposure to aspects of CT in problem solving. More importantly, the course addresses the gap in coding experiences among mathematics teachers in Singapore.

The course adopts a self-directed learning pedagogy, comprising 50% online asynchronous lessons where participants watch videos related to the week's topic(s) and attempt problem exercises (8 hours), and 50% online synchronous sessions where the class convenes to discuss the topic(s) and exercises (8 hours, for a total of 16 hours). The course also implements spiral learning, where key concepts in VBA and Excel are revisited in increasing difficulty as the course progresses [11]. Table 1 outlines the course topics, which include Excel formulas and VBA concepts and functions for solving/simulating specific mathematical problems. Course materials include slides, videos where the instructor demonstrates the use of Excel formulas and VBA functions, and problem exercises that require participants to apply these formulas and/or functions to solve/simulate mathematical problems. Only the videos and slides (featured in the videos) are discussed in this paper.

Table 1. Course content

Session	Topic(s) for that session
1	• Introduction to computational thinking in mathematical problem solving • Introduction to Excel layout, data types, and cell formatting • Basic Excel formulas (e.g., SUM(), AVERAGE(), SIN() etc.) • Using SOLVER tool to produce mathematical models
2	• Form controls (i.e., scroll bars) • Performing simulations for random number generation using RAND() and RANDBETWEEN() formulas • Condition checking using IF and COUNTIF formulas
3	• Creating more form controls (buttons) for running VBA code (assigning macro) • Introduction to VBA concepts (subroutines) and syntax (Sub, MsgBox, If...Then...ElseIf statement, logical and conditional operators (=, And, etc.)) • Declaring variables (Dim), assigning values to variables, variable name constraints, and data types (numeric and non-numeric) in VBA
4	• Performing iteration using For-Next loop statement in VBA • Creating user-defined functions via Function... End function statement in VBA • Condition checking using While-Wend loop statement in VBA
5	• Performing simulations for random number generation in VBA using Randomize and Rnd functions
6	

4.1 Developing CT habits through coding

This section examines implicit attempts in the course at developing CT habits through solving problems using computational tools including coding and flowcharts. An instructional video from Session 4 is presented to illustrate this point. As shown below in Figure 1, this video guides participants through the process of coding a “While loop” to compute the approximation of e^x using Taylor’s series.

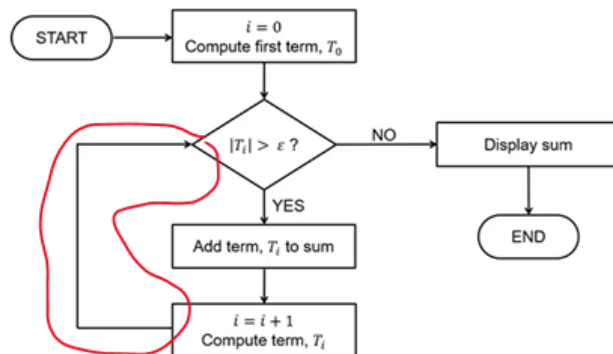
$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!} .$$

Mathematically speaking, the process of approximation is as follows (see above formula): beginning with a first term of 1, each successive term in the Taylor’s series expansion is computed and if the term is sufficiently significant (that is, large enough in magnitude to make a difference to the approximation), it is added to the partial sum. In coding, this process may be achieved computationally using a “While loop”. The following paragraphs explain how each step of coding a “While loop”, from a flowchart of simplified coding procedures to the actual VBA code, develops CT habits through encouraging certain coding practices/habits.

In the instructional video on the “While loop”, the instructor first introduces the basic syntax of the While loop statement (which checks for conditions and stops the (infinite) repetition/loop of a code when these conditions are met), before presenting a flowchart of the steps for computing the approximation of e^x using Taylor’s series, as described above. In the flowchart, the first term (T_0) is computed and compared with epsilon (ϵ) to determine if the condition of $|T_0| > \epsilon$ is fulfilled. If No, then the sum of terms (which contains only the value of T_0) will be displayed. If Yes, then T_0 is added to the sum of terms and the successive term (T_i) is computed and reevaluated against the condition of $|T_i| > \epsilon$ (where i represents the increase in degree for each successive term) to determine if the current sum of terms (containing only T_0) should be displayed or if T_i should be added to it. Terms will be continuously evaluated and added until the condition becomes “No”. The flowchart therefore incorporates the loop via branching conditions (YES / NO) and repetition/looping using arrows (see red highlight in (a) in Figure 1). By encouraging the habit of creating flowcharts to represent simplified coding procedures for solving a mathematical problem, this lesson helps participants to develop the CT habits of (a) algorithmic thinking through understanding the stepwise procedure of formulating solutions, (b) abstraction through extracting relevant information about Taylor’s series approximation for presentation in a flowchart, and (c) pattern recognition through understanding how the While loop can be applied not only in a general situation, but also in specific mathematical contexts (i.e., Taylor’s series). The course’s spiral learning approach is also demonstrated here, as the basic syntax and general usage of the loop is presented first, which scaffolds understanding of its application in more complex mathematical contexts later in the course such as in this example.

Next, the instructor shows how this flowchart can be converted into a full VBA code. Converting between different representations (i.e., flowchart to code) can be seen as helping participants develop the CT habit of abstraction, by identifying essential flowchart elements which need to be incorporated into a code so that it can perform the processes indicated in the flowchart. That is, by converting pertinent parts of the flowchart’s written procedures into code. One way this is done is through the coding habit of declaring variables to represent mathematical symbols in the flowchart. ϵ is declared as a variable called `epsilon` and the expansion of Taylor series as it increases in degree (T_i) is represented by variables like `taylor`, `term`, and `i`.

Example: Taylor Series again ...



(a)

VBA code

```

Sub TaylorSeries()
Dim i As Integer
Dim x, exact, taylor, term, epsilon As Double

epsilon = Cells(1, 2) ' read value of epsilon
x = Cells(2, 2) ' read value of x
taylor = 0 ' Initialise taylor to 0
term = 1 ' set value of first term
i = 0 ' initialize counter, i
While Abs(term) > epsilon ' check that term is large enough
    taylor = taylor + term ' add term to sum
    i = i + 1 ' increment counter, i
    term = (x ^ i) / factorial(i) ' find next term
Wend
exact = Exp(x)
Cells(4, 3) = exact
Cells(5, 3) = taylor
Cells(8, 2) = i - 1
End Sub
  
```

(b)

Figure 1. The “While loop” instructional video in Session 4

The instructor then sequentially examines each line/sequence of lines in the code and explains their purpose with respect to solving the problem (see (b) in Figure 1). Participants are encouraged to develop the habits of dealing with code sequentially and seeing code as a combination of chunks/modules with different functions that work together to solve an overarching problem. For example, the instructor first explains how the line `taylor = 0` and `i = 0` initialises the variables

taylor (sum of terms) and i (increase in degree for each successive term) respectively to 0, and how `term = 1` sets the value of the first term. He then moves down the code to clarify that the use of the While loop in `While Abs(term) > epsilon` checks for the condition of whether $|T_i|$ is more than ϵ . If $|T_i| > \epsilon$, then the term is added to the sum (`taylor = taylor + term`), and the degree (`i = i + 1`) and the value (`term = (x ^ i) / factorial(i)`) of the successive term is computed and compared with ϵ in a new loop. This helps participants to cultivate the CT habits of (a) algorithmic thinking by understanding the sequential execution of codes, and (b) decomposition by breaking down codes into modules and understanding how each module works.

5. Analysis of course feedback

5.1 Quantitative findings

Participants' feedback was sought through a form that contains statements about the course. Table 2 presents these statements, which are grouped into five categories, and participants had to indicate their level of agreement (in a typical 5-point Likert-scale) with these statements. For easier reference, these statements were coded (e.g. IT1, CLO2 etc.) based on the category names.

Table 2. Statements in the Likert-scale-based feedback

Category	Statement
<i>Instructor's teaching</i> (IT)	IT1: The facilitator is skillful at facilitating the participants' learning.
	IT2: The presentation was clear.
<i>Course learning outcomes</i> (CLO)	CLO1: The objectives were achieved.
	CLO2: The course helped me to reflect on my teaching practices.
	CLO3: The course met my learning needs.
	CLO4: I can apply the ideas/knowledge/skills learnt from the course.
	CLO5: The questions were adequately addressed at the course.
<i>Course materials</i> (CM)	CM1: The learning resources supported me in my learning.
<i>Likeness to recommend this course</i> (LRC)	LRC1: I would recommend the course/workshop to my colleagues.
<i>Course structure</i> (CS)	CS1: The duration of the course was sufficient to meet its objectives.*

*CS1 is excluded from Figure 2 as it uses a different rating scale ("Just right" and "Too short")

Figure 2 below shows participants' ratings for statements across the five categories. Generally, more than 90% of participants expressed agreement ("Strongly Agree" or "Agree") with the statements. Certain statements received 100% agreement, and these are related to: the instructor's skillfulness at teaching (IT1), addressing of participants' questions (CLO5), and learning support provided by the course's learning resources (CM1). Only a minority (4 to 18%) disagreed ("Disagree" or "Strongly Disagree") with a few statements concerning: the clarity of instruction (IT2), attainment of course objectives (CLO1), impact on participants' teaching practices (CLO2), addressing of participants' learning needs (CLO3), applicability of course content (CLO4), course duration (CS1), and likeliness to recommend the course to others (LRC1). All ten statements reflect the strengths of the course due to high agreement ratings, although the presence of negative ratings in some of them suggests that these are aspects of the course that can be improved on.

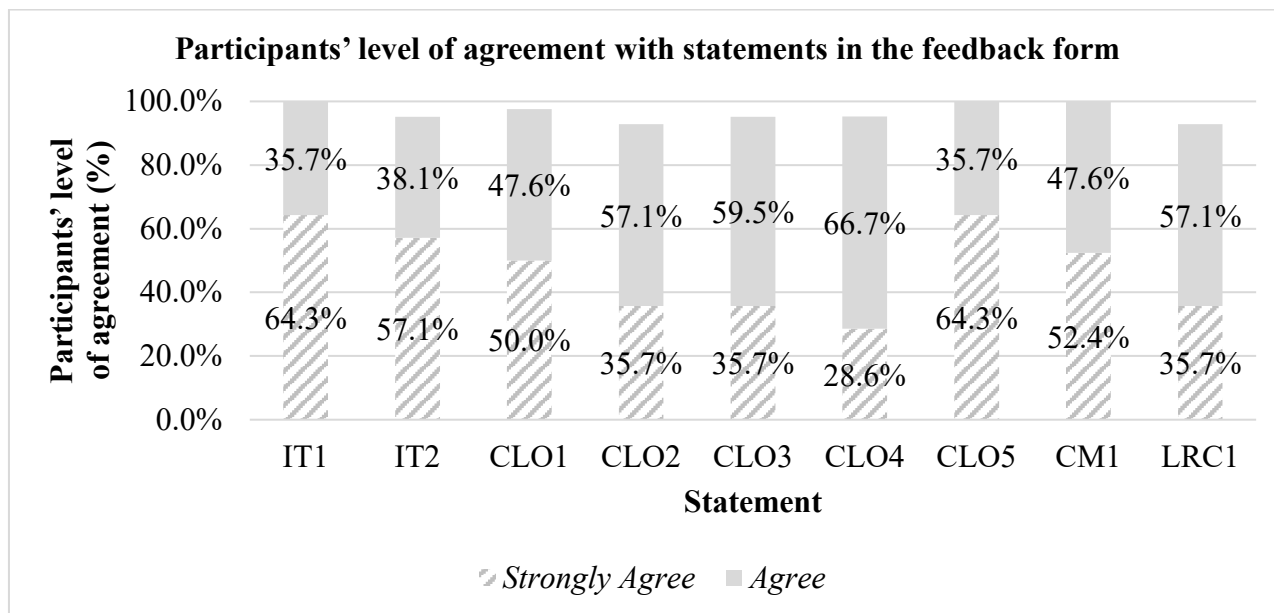


Figure 2. Participants' level of agreement with statements in the Likert-scale-based feedback

5.2 Qualitative findings

Participants' responses to open-ended questions were analysed and grouped into three main themes, namely, (1) course delivery, (2) course difficulty, and (3) usefulness and applicability of the course content. Their responses to these questions form the basis of the qualitative findings for this study. A total of 21 subthemes were uncovered. However, due to space constraints, only the nine most prominent subthemes are discussed here. These nine are the top three to four subthemes per theme except for "course difficulty" which has only two subthemes (see Table 3). The content of these responses, categorised as strengths and areas for improvement, are summarised in Table 3, while notable responses are cited in the analysis below.

The theme of usefulness and applicability of the course content dominates, followed by course delivery and course difficulty. For the theme of usefulness and applicability of course content, prevalent subthemes are related to whether the course addresses the curricular/pedagogical needs of mathematics educators, how it exposes participating educators to various uses of VBA, and the educators' general learning gains. Next, the allocation of learning/teaching activities, the course's/instructor's level of engagement, clarity of the instructor's teaching, and pacing of the course are prominent subthemes associated with the theme of course delivery. Lastly, the theme of course difficulty comprises two subthemes: the course content and instructor's teaching. These subthemes suggest the main strengths of the course as well as pertinent areas of concern that can be addressed in future course runs or other PD courses on coding.

Next, the specifics of strengths and areas for improvement in each theme are presented. In relation to the theme of usefulness and applicability of the course content, many participants described useful ideas from the course that they could apply to their mathematics lessons, such as the use of simulations to help students visualise concepts in topics like statistics as well as creating activities that develop students' CT. Some described how they have acquired the skills to implement

these ideas, and others anticipated some learning benefits for students such as deepening metacognition, simplifying concepts, and developing CT.

Table 3. Themes and subthemes in the open-ended feedback

Theme	Subtheme	Summary of strengths	Summary of areas for improvement
Usefulness and applicability of the course content	<i>Addressing mathematics curricular/ pedagogical needs</i>	• Promotes awareness of the pedagogical potential of VBA in mathematics and its benefits • Shows how to create pedagogical activities for mathematics using these tools	• Provide examples tailored to participating educators' curricular needs • Provide examples of how VBA is used in classrooms globally
	<i>Exposure to different uses of VBA</i>	• Promotes awareness of diverse uses of VBA	• Introduce more complex applications of VBA
	<i>General learning gains</i>	• Perceived gains in pedagogical knowledge and learning of VBA/programming	
Course delivery	<i>Allocation of course activities</i>		• More hands-on coding practice • More time to discuss solutions and do exercises • Focus more on covering content in synchronous lessons, less on flipped classroom and asynchronous lessons
	<i>Level of engagement</i>	• The instructor is passionate, approachable, addresses questions, open to sharing his expertise, and creates fun exercises	• Engage more with participants through questions and coding demonstrations
	<i>Clarity of instruction</i>	• The instructor is clear and precise at explaining concepts	
	<i>Course pacing</i>	• The instructor is patient • Pace of the course is ideal and tailored to address queries and participants' differentiated speed of coding	• Course is slow paced and draggy, especially for synchronous lessons
Course difficulty	<i>Course content</i>	• The difficulty level of exercises is ideal	• Provide hints for the exercises • Set less exercises as homework • Exercises are difficult, especially for those with no coding experience • Exercises should be simplified and/or more scaffolded/guided

			• Exercises can be more difficult • More time for exercises
	<i>Instructor's teaching</i>	• The instructor makes learning easy	• In the synchronous lessons, account for those with no coding experience

One participant reported gains in coding skills through the course, being able to now “*create some simple codes and students can interact [with these codes] to understand some concepts better*”, while another became aware of how Excel could positively impact students’ mathematics learning by “*facilitat[ing] learning of mathematics to deepen students’ metacognition*”. Participating teachers also expressed that they gained an understanding of how VBA can be used in diverse contexts like administrative work, general simulations, and results analysis. One of them was “*impressed by what excel VBA can achieve*”, while another reflected on his/her already ongoing use of VBA in “*daily productivity work... with Excel by automation*”.

Several respondents also discussed their overall learning gains, such as how they received pedagogical guidance and the enjoyment of learning coding in the course. They described the instructor as “*great at guiding [them] in [their] thinking and pedagogy*” and an exemplary figure for “*teachers in designing lessons for blended learning*”, adding that the process of learning VBA was “*meaningful to do*” and became a stimulus “*to explore a bit more of programming*”.

Nevertheless, participating teachers also noted some aspects of the course which could be improved regarding its utility and applicability. There were a few suggestions for more problem exercises and examples that address the educators’ own curriculum/syllabus (i.e., A-level mathematics) as well as real-life instances of VBA usage in mathematics instructional contexts to “*seed more ideas for [teachers] to see the possibilities*” of how they can apply it to their own classes, because mathematics teachers in Singapore currently have, as one educator perceives, “*insufficient triggers as [they] still lack the exposure*”. One teacher also suggested a further exploration of more complex VBA applications such as in animations.

In terms of course delivery, participants generally felt that the instructor was able to explain concepts with clarity and adjust his pace to adapt to their questions and differing speeds of learning. They also found that they were able to clarify their doubts, and some enjoyed the exercises. However, participants also highlighted several areas that could be improved. They pointed out that more time could be allocated for course activities, especially for attempting the exercises and for discussion of the solutions to these exercises. Quite a few thought that coding could be practiced during lessons, rather than as part of the asynchronous activities. In fact, some expressed that the focus on the flipped classroom approach could be reduced and instead more content could be covered in synchronous lessons.

A small number of teachers also proposed that apart from encouraging queries, live coding demonstrations would be “*most useful instead of just videos*” in boosting engagement while a few others remarked that the course, especially in its synchronous sessions, was too slow-paced and draggy, with one recommending that the course summary “*could outline in more detail what the course will cover*” prior to participants registering.

Lastly, in terms of course difficulty, a few participants commented that the exercises were of “*the right level of difficulty*” while some underscored the instructor’s teaching skills in making the course “*easy to follow*” and “*comfortable*” especially for the asynchronous sessions. On the flip side, there were numerous suggestions to reassess the difficulty level of the course content. Many teachers voiced concerns about the difficulty level of the exercises, mentioning that these were “*quite daunting*”.

for newcomers who are not familiar with coding". They provided several suggestions to alleviate this, such as more hints or guidance/scaffolding from the instructor, simpler exercises, reducing the number of exercises, and increasing the time given for solving these exercises. Comments that proposed more hints and/or scaffolding appeared several times, remarking that due to the lack of these aids, such as the intentional omission of certain VBA functions, it *"was quite difficult to figure out what needs to be used sometimes for the exercises"* and *"the math involved [in VBA/coding] can be unfamiliar"*. On the contrary, one respondent proposed that the difficulty level could be increased. Pertaining to the instructor's teaching, one participant expressed that during synchronous sessions, some teachers *"without coding background may struggle to understand"* immediately what is being taught in the class itself.

6. Discussion and implications

In general, the above findings show that the course presented in this study can potentially help learners, including practicing mathematics teachers, develop some level of coding skills through designing and implementing computational solutions to mathematical problems. Secondly, while the course displays strengths in many aspects such as course materials and instructor's teaching, there are some areas for improvement, especially in aspects like course learning outcomes and level of course difficulty. Nevertheless, these findings suggest the role of coding activities in developing a computational mindset and the applicability of these activities in mathematics learning.

Furthermore, findings from this study also suggest implications for pedagogical principles and practices. One should bear in mind the importance of accommodating variations in coding aptitude and experience amongst mathematics teachers, and perhaps a course with differentiated instruction could be designed. Coding examples could be better prepared with attention paid to tailor them to teachers' curricular/pedagogical needs. The feedback analysis revealed that a number of teachers expressed difficulty in handling the exercises, especially for those without coding experience, and several suggested exercises that were more relevant to their curricular requirements.

Some were also not very comfortable with the self-directed/flipped classroom approach and instead preferred more guided learning. Offering differentiated guidance and targeted examples helps educators see the value and direct applicability of CT and coding in their teaching and ensures that educators with different coding abilities can receive customised guidance, rather than being demotivated to adopt coding and CT due to perceived difficulty and/or lack of relevance.

In terms of policies, analysing intended learning outcomes in the course materials and perceived learning gains in the feedback reveals potential mismatches between mathematics teachers' beliefs and educational policies regarding CT. Although the findings suggest that the course materials could be effective in cultivating CT and coding skills and that the course exhibits strengths in doing so, there are numerous areas of concerns that still need to be addressed. These include the relevance of the course content to the educators' pedagogical/curricular needs and their perceptions of the feasibility of acquiring coding skills with regards to factors like the difficulty of coding, amount of guidance received, and preferences for certain pedagogical approaches. Educational policies concerning PD that involves coding or CT should be informed by studies on what mathematics educators need, prefer, and are comfortable with in their teaching and learning as well as what has been done well and what needs to be improved in existing PD courses and teaching resources.

Lastly, in terms of research, this study serves as a starting point for piloting and evaluating PD courses or resources related to coding and CT for mathematics teachers, given the lack of such programmes and information about their impact, strengths, and areas for improvement especially in Singapore.

7. Conclusion

This study evaluated the potential impact of a PD course that focusses on computational approaches in mathematical problem solving, with a view towards developing coding competencies and CT habits in the context of mathematics learning. This was done by analysing the course's strengths and areas for improvement in various aspects of the course, and examining how an instructional segment helps learners develop some aspects of CT by solving a problem computationally.

There are some limitations that may affect the generalisability of the findings. Firstly, the course has a short duration of only six sessions. It has been suggested that a short course duration may not be very effective for developing teachers' CT in the long run [24, 32]. Secondly, the course was only taught thrice and had a relatively small participant size (average of 14 per run). More course runs with larger participant numbers may be necessary to verify the effect and usefulness of such a PD course. Lastly, the analysis only reflects intended learning objectives (in the materials) and participants' perceived learning outcomes. Classroom footage or CT scores would provide more concrete evidence of learning gains.

Nevertheless, despite these limitations, insights gained from this study will serve as a stepping stone to developing more effective and better designed PD courses and teaching resources on coding and CT. In fact, the present study has provided some very valuable lessons and directions for the authors' next research project on mathematics teachers' CT knowledge and their views on CT integration in mathematics instruction, its challenges, and CT pedagogical tools.

Acknowledgements: The authors would like to thank National Institute of Education for their generous financial support (Grant Award No. 023245-00001; Funder Reference No. RS 9/23 AKC)

References

- [1] Ang, K. C. (2019). *Mathematical modelling for teachers: Resources, pedagogy and practice*. New York: Routledge.
- [2] Ang, K. C. (2020). Computational thinking as habits of mind for mathematical modelling. In W.-C. Yang & D. Meade (Eds.), *Proceedings of the 25th Asian Technology Conference in Mathematics* (pp. 126-137). Radford: Mathematics and Technology, LLC.
- [3] Ang, K. C. (2021). Computational thinking and mathematical modelling. In F. K. S. Leung, G. A. Stillman, G. Kaiser, & K. L. Wong (Eds.), *Mathematical modelling education in East and West* (pp. 19-34). Cham: Springer.
- [4] Araujo, R. C., Floyd, L., & Gadanidis, G. (2019). Teacher candidates' key understandings about computational thinking in mathematics and science education. *Journal of Computers in Mathematics and Science Teaching*, 38(3), 205-229.
- [5] Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2), 77-101.
- [6] Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*.
- [7] Chan, S.-W., Looi, C.-K., Ho, W. K., Huang, W., Seow, P., & Wu, L. (2021). Learning number patterns through computational thinking activities: A Rasch model analysis. *Heliyon*, 7(9), 1-14.

- [8] Chan, S.-W., Looi, C.-K., Ho, W. K., & Kim, M. S. (2023). Tools and approaches for integrating computational thinking and mathematics: A scoping review of current empirical studies. *Journal of Educational Computing Research*, 60(8), 2036-2080.
- [9] Csernoch, M., Biró, P., & Máth, J. (2021). Developing computational thinking skills with algorithm-driven spreadsheets. *IEEE Access*, 9, 153943-153959.
- [10] Demir, K., Çaka, C., Yaman, N. D., İslamoğlu, H., & Kuzu, A. (2022). Examining the current definitions of computational thinking. In Information Resources Management Association (Ed.), *Research anthology on computational thinking, programming, and robotics in the classroom* (pp. 224-252). Hershey: IGI Global.
- [11] Ernest, P. (2001). Spiral learning. In L. S. Grinstein & S. J. Lipsey (Eds.), *Encyclopedia of mathematics education* (pp. 669-670). London: Routledge.
- [12] Gadanidis, G., Cendros, R., Floyd, L., & Namukasa, I. (2017). Computational thinking in mathematics teacher education. *Contemporary Issues in Technology and Teacher Education*, 17(4), 458-477.
- [13] Ho, W. K., & Ang, K. C. (2015). Developing computational thinking through coding. In W.-C. Yang, D. Meade, & C. Liu (Eds.), *Proceedings of the 20th Asian Technology Conference in Mathematics* (pp. 73-87). Radford: Mathematics and Technology, LLC.
- [14] Ho, W. K., Ghosh, J., Kaur, B., & Looi, C.-K. (2022). Spreadsheets: A technological nexus between big ideas in mathematics and computational thinking in mathematics classrooms. In W.-C. Yang, M. Majewski, D. Meade, & W. K. Ho (Eds.), *Proceedings of the 27th Asian Technology Conference in Mathematics*. Radford: Mathematics and Technology, LLC.
- [15] Humble, N., & Mozeliuss, P. (2023). Grades 7-12 teachers' perception of computational thinking for mathematics and technology. *Frontiers in Education*, 8, 1-13.
- [16] Kannadass, P., Hidayat, R., Siregar, P. S., & Husain, A. P. (2023). Relationship between computational and critical thinking towards modelling competency among pre-service mathematics teachers. *TEM Journal*, 12(3), 1370-1382.
- [17] Kong, S.-C. (2019). Learning composite and prime numbers through developing an app: An example of computational thinking development through primary mathematics learning. In S.-C. Kong & H. Abelson (Eds.), *Computational thinking education* (pp. 145-166). Singapore: Springer.
- [18] Kong, S.-C., Abelson, H., & Kwok, W.-Y. (2022). Introduction to computational thinking education in K-12. In S.-C. Kong & H. Abelson (Eds.), *Computational thinking education in K-12: Artificial intelligence literacy and physical computing* (pp. 1-12). London: MIT Press.
- [19] Kong, S.-C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers & Education*, 151, 1-19.
- [20] Liu, H. P., Perera, S. M., & Klein, J. W. (2017). Using model-based learning to promote computational thinking education. In P. J. Rich & C. B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 153-172). Cham: Springer.
- [21] Martins, S. (2022). Computational thinking and mathematics learning: An experience in a pre-service teachers course. In L. G. Chova, A. L. Martínez, & J. Lees (Eds.), *EDULEARN22 Proceedings: 14th International Conference on Education and New Learning Technologies* (pp. 6831-6840). Valencia: IATED.
- [22] Miller, J. (2019). STEM education in the primary years to support mathematical thinking: Using coding to identify mathematical structures and patterns. *ZDM*, 51(6), 915-927.
- [23] Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

- [24] Sands, P., Yadav, A., & Good, J. (2018). Computational thinking in K-12: In-service teacher perceptions of computational thinking. In M. S. Khine (Ed.), *Computational thinking in the STEM disciplines: Foundations and research highlights* (pp. 151-164). Cham: Springer.
- [25] Sanford, J. F. (2018). Introducing computational thinking through spreadsheets. In M. S. Khine (Ed.), *Computational thinking in the STEM disciplines: Foundations and research highlights* (pp. 99-124). Cham: Springer.
- [26] Sanford, J. F., & Naidu, J. T. (2017). Mathematical modeling and computational thinking. *Contemporary Issues in Education Research*, 10(2), 159-168.
- [27] Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, 18(2), 351-380.
- [28] Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Exploring the science Framework and NGSS: Computational thinking in the science classroom. *Science Scope*, 38(3), 10-15.
- [29] van Borkulo, S. P., Chytas, C., Drijvers, P., Barendsen, E., & Tolboom, J. (2023). Spreadsheets in secondary school statistics education: Using authentic data for computational thinking. *Digital Experiences in Mathematics Education*, 9(3), 420-443.
- [30] 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.
- [31] Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.
- [32] Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). Computational thinking in teacher education. In P. J. Rich & C. B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 205-220). Cham: Springer.