

# Chapter 11

## Undergraduate Mathematics Students Engaging in Problem-Solving Through Computational Thinking and Programming: A Case Study



Said Hadjerrouit and Nils-Kristian Hansen

### 11.1 Introduction

Students in mathematics study programs are expected to acquire basic algorithmic and computational thinking (CT) skills, in addition to learning emerging programming languages (Wing, 2014). These are considered important competencies for future work in society and should be acquired by all university mathematics students to improve their mathematical problem-solving skills by benefitting from CT and the power of programming languages (Shute, Sun, & Asbell-Clarke, 2017). It is assumed that with easier access to digital technology in higher education, the integration of programming activities and application of CT skills into teaching and learning could be easier than in previous years. Indeed, until the relatively recent re-emergence of interest for integrating CT and programming in school mathematics, this topic received relatively little attention in the research community. Still, despite the renewed interest, there is little research on linking CT and programming with mathematics at the university level (Buteau, Muller, Mgombelo, Sacristán, & Dreise, 2020).

This study is an attempt to *contribute* toward the advancement of the potential value of CT at the undergraduate level when students engage in mathematical problem-solving through CT and programming, while uncovering both opportunities and challenges encountered by the students when trying to connect mathematics with CT and programming.

This chapter is structured as follows. Firstly, the theoretical framework is outlined. Secondly, the chapter describes the context of the study, research question, methods, and the mathematical task to be solved. Then, the results are reported and

---

S. Hadjerrouit (✉) · N.-K. Hansen  
University of Agder, Kristiansand, Norway  
e-mail: [Said.Hadjerrouit@uia.no](mailto:Said.Hadjerrouit@uia.no)

analyzed. This is followed by a discussion and pedagogical implications of the results. Finally, the limitations of the study are highlighted, and recommendations for future work conclude the chapter.

## 11.2 Theoretical Framework

The application of computational thinking (CT) and programming for mathematical problem-solving has had a relatively long history in mathematics education (Papert & Harel, 1991; Shodiev, 2015). Today, CT and programming languages are becoming a key learning goal of mathematics courses from primary to university level. Drawing on the research literature, the theoretical framework and associated terms and notions underlying this study are outlined in the following sections.

### 11.2.1 *Computational Thinking (CT): A Review of the Research Literature*

While a large volume of research studies on CT and programming in mathematics at the school level exists, there is little research at the university level on linking CT and programming with mathematics. Some important research issues and challenges have been identified.

Broley, Caron, and Saint-Aubin (2018) reported that teachers have different understanding and interpretations of CT and programming. They do not associate it automatically with a systematic and well-organized way of mathematical problem-solving with best possible outcome. Moreover, there may be numerous justifications for exposing students to CT and programming depending on the educational situation.

Similarly, Li et al. (2020a, 2020b) argued for the complexity of CT, which is not synonym with “computing” or “computer” in a restricted sense. Rather, CT is a model of thinking with a multifaceted theoretical perspective. CT is important not only in computer science and mathematics but also in other disciplines of STEM.

In more general terms, Malyn-Smith and Angeli (2020) distinguished between two definitions of CT, one focusing on defining CT by disaggregating its elements and another on exploring the integration of CT into disciplinary learning through its practices. While there are similarities between these two definitions, there are several differences. Accordingly, the challenge includes the evolution of a common definition of CT, and a shared understanding of CT, comprehensively integrating CT into curricula.

In terms of theoretical frameworks, Buteau, Muller, Marshall, Sacristán, and Mgombelo (2016) discussed CT and programming from a broad perspective based on Wing (2008) and other researchers, and from the perspective of mathematicians’

research practices, partly informed by the constructionist paradigm (Papert & Harel, 1991), before they developed their theoretical framework on their view of learning mathematics by engaging in CT, drawing mainly on the instrumental approach.

Similarly, Buteau et al. (2018) used the constructionist approach for the classroom implementation of programming and the instrumental approach to address a student's appropriation of CT and programming as a tool for the exploration of a mathematics concept, theorem, conjecture, or a real-world situation. See also Buteau et al. (2020) for the use of instrumental genesis (an important element of the instrumental approach) stages of programming for mathematical work.

Gueudet et al. (2020) also used the instrumental approach to highlight the links and connections between mathematical and programming competencies, which are quite complex and increasingly important at the university level. The study recommends deepening the knowledge about these complex links and their evolution, including CT even though this notion is not mentioned in the article.

In line with Gueudet et al. (2020), an overview of research done by DeJarnette (2019) concluded that the question that is still underdeveloped in existing literature on CT at the university level is how students develop skills when interacting within an environment that merges and connects mathematical thinking with CT and programming.

Summarizing, the research literature reveals important issues and challenges to be addressed for the advancement of mathematical problem-solving through CT and programming and the importance of integrating these skills into mathematics courses and work of today's mathematician (Broley et al., 2018). This study aims to address some of these issues and challenges, in particular the complex links between mathematics, CT, and programming.

### ***11.2.2 Computational Thinking (CT) and Mathematical Thinking***

Several definitions of the term CT exist in the research literature. Wing (2014) described CT as “the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out.” Misfeldt and Ejsing-Duun (2015) described CT in similar words. It is the ability to work with algorithms understood as systematic and structured descriptions of problem-solving and construction strategies. Filho and Mercat (2018) defined algorithmic thinking as the process of solving a problem step by step in an effective, nonambiguous, and organized way that can be translated into instructions to solve problems of the same type by an individual or a computer.

Wing (2008, 2014) indicated that the main commonality between CT and mathematical thinking is problem-solving and a structured step-by-step construction process. Likewise, mathematical thinking consists of problem-solving processes, beliefs about mathematics, and justification for solutions (Shute et al., 2017, p. 145).

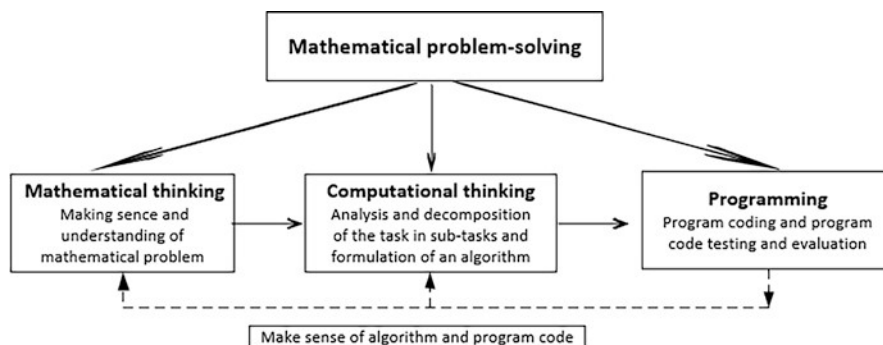
Accordingly, it involves the application of mathematical skills to solve mathematical problems, e.g., equations and functions (Shute et al., 2017, p. 145). The authors concluded that mathematical thinking and CT have a lot of communalities: problem-solving, modeling, data analysis and interpretation, as well as statistics and probability (Shute et al., 2017, p. 145). Furthermore, CT has communalities with engineering thinking in terms of design and evaluation of processes (Pérez-Marín, Hijón-Neira, Bacelo, & Pizarro, 2020), which is similar to algorithmic thinking and design of programming code. Moreover, CT and programming constructs such as variables and flow statements (if-then-else, for, while, repeat, etc.) are closely connected to arithmetical and mathematical thinking (Lie, Hauge, & Meaney, 2017).

This close connection between mathematical thinking and CT might provide opportunities for mathematical problem-solving. In contrast, programming skills alone without the mediation of CT are important but may not be sufficient to improve mathematical problem-solving. Clearly, CT is not the same as programming, but being able to program and test program codes is a result of being able to think computationally (Li et al., 2020a, 2020b; Shute et al., 2017; Wing, 2008). Thus, CT skills are critical for building efficient algorithms for mathematical problem-solving rather than trial and error and getting the program to run (Topalli & Cagiltay, 2018). In other words, CT requires students to be engaged in a continuously changing problem-solving process until an appropriate solution is found by designing effective algorithms that can be translated into computer programs.

### ***11.2.3 A Three-Step Iterative Approach to Mathematical Problem-Solving Through CT and Programming***

Drawing on the research literature (Kotsopoulos, Floyd, Nelson, Makosz, & Senger, 2019; Lee & Malyn-Smith, 2020; Romero, Lepage, & Lille, 2017; Santos, Tedesco, Borba, & Brito, 2020; Weintrop et al., 2016), this chapter proposes a three-step iterative approach to connect mathematical problem-solving with CT and programming (Fig. 11.1).

Firstly, students should have a good mathematical background to benefit from CT and programming languages. More specifically, they should be able to benefit from their knowledge to make sense of the task and have a mathematical understanding of it before formulating an algorithm and starting programming. Secondly, CT should enable students to analyze and decompose the mathematical task into smaller sub-tasks, analyze them in a different way than one would otherwise do in educational settings, and design an algorithm on how to solve it step by step before programming it. Engaging students in mathematical problem-solving through CT may enable a better understanding of mathematics beyond textbook mathematics and paper-pencil techniques. Thirdly, students should be able to translate the mathematical solution with the associated algorithm to the constructs of the programming language. This presupposes that the language is usable for novice students.



**Fig. 11.1** Three-step approach to connect mathematical problem-solving with CT and programming

This is not a linear approach beginning with a mathematical problem and ending with programming and code testing. The approach may include numerous feedbacks to previous steps to make sense of the algorithm and program output. It is also particularly important to consider interactions between mathematical thinking and CT. Using CT and programming in mathematics courses should provide opportunities to help students do mathematics and gain new knowledge that is otherwise difficult to acquire without experimenting with the program and thinking algorithmically. However, this might be difficult to achieve for novice students unless the mathematical tasks are well designed, mathematically sound, and faithful to the underlying mathematical properties.

### 11.2.4 Programming and Usability Issues

When referring to the notion of usability or usable technologies in mathematics education, the research literature focuses on educational software such as GeoGebra, CAS, and SimReal (Artigue et al., 2009; Bokhove & Drijvers, 2010; Hadjerrouit & Gautestad, 2019). However, programming languages differ from educational software and how they are used to implement mathematical problems where one, for example, can graph a function or compute an integral simply by entering the function and pressing a button in GeoGebra or similar. Hence, evaluating the usability of programming languages might not be as straightforward as it may seem. Still, three usability criteria can be applied to programming languages with slight modification.

The first criterion is the degree to which the user interface of the programming language is easy to use and understand. Secondly, a usable programming language should allow a quick familiarization with it in terms of learning the language constructs, such as variables, if-then-else, for and while loop, or repeat. The third criterion is the quality of feedback provided by the program in terms of semantic and

syntax error messages and whether these are useful to foster a successful implementation of the algorithmic solution through testing, correcting and improving the program, and making sense of the program output in terms of the mathematical solution to the problem.

These usability issues are connected to each other. Interface usability and familiarization with the programming language constructs are a prerequisite for successful implementation of the program and making sense of the solution. Indeed, technical programming constraints may result in demotivation and frustration from using the programming language. This can happen if there are technical usability issues or the program is not well designed. It can then be difficult to detect runtime or syntax errors, even if the programming language being used—MATLAB—comes up with hints. As a result, students may not work on their own if the feedbacks from the program are not comprehensive.

### ***11.2.5 CT, Programming, and Pedagogical Considerations***

A purely technical approach to CT and programming will not succeed unless students' engagement with mathematical problem-solving is placed in a pedagogical context. A pedagogy-based approach to CT and programming should enable a good degree of autonomy so that the students can work on their own and have a sense of control over their mathematical learning. Clearly, students should be able to acquire knowledge without being completely dependent on the teacher. Moreover, CT and programming languages should be a motivational factor for learning mathematics. In other words, CT should support students' engagement with problem-solving by means of intrinsically motivating tasks that are tied to the students' mathematical activities. Finally, interacting with a programming language when engaging in mathematical problem-solving should be mediated by CT through a structured construction of effective algorithms.

## **11.3 The Study**

### ***11.3.1 Context of the Study and Research Question***

This work is a single case study conducted in the context of a first-year undergraduate course on programming with applications in mathematics. The participants were two students volunteering from a class of 8, enrolled in the course in 2019. The students had varied knowledge background in mathematics, but no experience with the programming language MATLAB. The course introduced the basic constructs of MATLAB, e.g., single variables, arrays, control flow statements, and functions. The course also discussed the major steps in systems development, i.e., analysis,

design, implementation, and testing. Ultimately programming was used for numerical analysis. No explicit training on CT was provided, though. Student programming exercises made up a major part of the course. However, they had no focus on CT.

The research question addressed in this chapter is: *How do students engage in mathematical problem-solving through CT and programming activities?*

### 11.3.2 Mathematical Task

The mathematical task presented to the students in this research study was as follows: “The length of a curve may be approximated using Pythagoras’ theorem by positioning a triangle adjacent to the curve (Fig. 11.2, left, below). The length of the green line between A and B may then be approximated as  $\sqrt{x^2 + y^2}$ . The task is to write a MATLAB function approximating the curve length of  $f(x) = 2^x$  between two given  $x$ -values (Fig. 11.2, right, below).

A function skeleton is as follows:

- Function length = length\_estimate( $x_1, x_2$ ),
- Length =?;

Determine the formula, based on  $x_1$  and  $x_2$ , to replace the question mark. To calculate the square root, you may use the MATLAB function sqrt. The command sqrt( $x$ ) will produce  $\sqrt{x}$ .”

Approximating curve length in this fashion is a mathematically sound first approach. If extended to a sum of an ever-increasing number of ever smaller triangles, it will converge to the actual curve length.

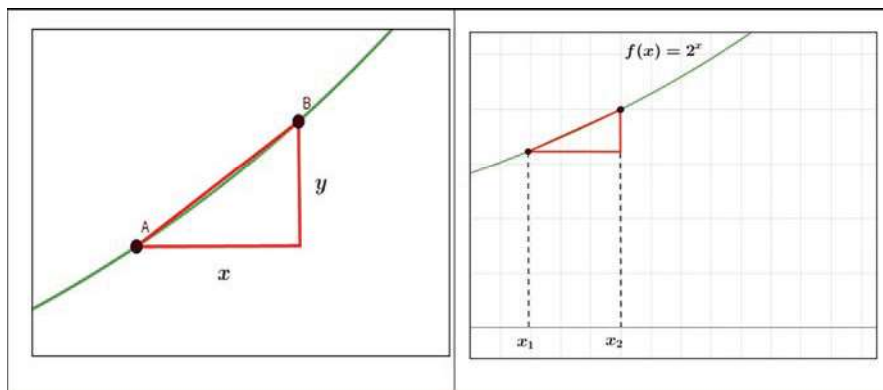


Fig. 11.2 The mathematical task

### 11.3.3 The Programming Language MATLAB

MATLAB is a mathematics software with a built-in C-based scripting language. A screenshot of the script editor containing a suggested solution to the task is shown in Fig. 11.3.

This is very similar to C, except for that in MATLAB syntax the function return variable, e.g., “length” in Fig. 11.3, is located in the function header instead of in a separate return statement.

### 11.3.4 Data Collection and Analysis Method

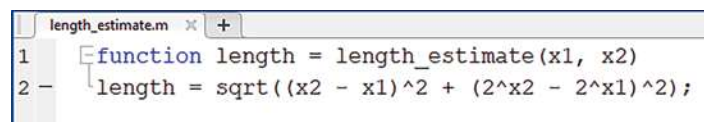
The two participating students were presented with a mathematical task to solve, while responding to questions in a dialogue with the teacher on the mathematical solving process. The main data collection method used is a task-based semi-structured interview with the students. Open-ended questions were also used to gain a deeper understanding of some important issues.

In terms of data analysis of the results, the three-step approach presented in the theoretical framework (Sect. 2.3) served as a reference for analyzing the students’ problem-solving process, that is:

1. Understand the mathematical task.
2. Analyze and decompose the mathematical task, and then design an algorithm on how to perform the solution step by step before programming it.
3. Finally, translate the algorithmic solution to the MATLAB programming language code, with eventual feedbacks to previous steps 1 and 2.

More specifically, the students were expected to solve the mathematical task presented to them in Sect. 3.2. in three steps as follows:

- Understand the task, that is, using Pythagoras’ theorem to calculate the hypotenuse.
- Formulate an algorithm, that is, find the lengths of the triangle hypotenuse using the function  $f(x) = 2^x$ , relating it to  $x_1$  and  $x_2$ .
- Translate the algorithm into MATLAB code, corresponding to the sample shown in Fig. 11.3.



```
length_estimate.m x +
1 function length = length_estimate(x1, x2)
2     length = sqrt((x2 - x1)^2 + (2^x2 - 2^x1)^2);
```

Fig. 11.3 MATLAB script editor with task solution



The analysis of the results seeks indications of students' problem-solving through CT by means of MATLAB according to this three-step approach. This is not the same as analyzing and coding in the sense of grounded theory without theoretical background. Rather, it is an analytical tool that tries to address the research question about how students engage in mathematical problem-solving through CT and programming activities drawing on students' interviews when solving the mathematical task. The interviews were transcribed and analyzed according to an inductive strategy based on the interplay between the three-step approach and the empirical data collected by means of semi-structured interviews (Patton, 2002).

## 11.4 Results

The results describe how the participating students engage in mathematical problem-solving through CT and MATLAB. The students were given the task described in Sect. 3.2. The abbreviations S1 and S2 are used for the students, and T for the teacher.

### 11.4.1 Student 1 (S1)

At the time of the interview, the student almost had completed the first semester of a bachelor program in mathematics. The interview took place in a classroom with the student, the teacher, and an observer present. Audio from the session was recorded and the dialogue later transcribed. The student brought own computer and used it to program in MATLAB. A piece of paper and a pen was put in front of the student with instructions to use if desired. The task was presented to the student on a sheet of paper as the interview started.

It took some time before the student made sense of the mathematical task. The teacher then asked the student to develop a skeleton of the solution. The student did, and started thinking about the length, but suggested an incorrect solution, based only on the values of  $x_1$  and  $x_2$ :

*S1: I first sat and thought function length, estimating length,  $x_1$ ,  $x_2$  and then I thought length is equal to the square root of  $x_1$  to the power of two plus  $x_2$  to the power of two.*

At this stage, the student showed a basic mathematical and algorithmic understanding of the task, by referring to the mechanisms of Pythagoras' theorem and linking it to the variables available in the MATLAB function. However, the student failed to see the need for including the function  $f(x) = x^2$  in the calculations.

After some calculation trials, the student noticed that the attempted solution was wrong. Then, the teacher encouraged the student to think computationally:

T: *But before you start using MATLAB, are you going to make an algorithm (...), for problem-solving before you start using MATLAB?*

S1: *I just have to sit and think about it.*

But still, the student continued guessing and calculating without thinking computationally. Neither did the student attempt to make use of pen and paper as an aid in structuring the thought processes, for instance, by drawing a sketch. After a while, the student's mental model reversed, and the student attempted to calculate the known  $x$  and  $y$  coordinates from the unknown hypotenuse. At this point the teacher intervened:

T: *Now you say you know the hypotenuse and calculate  $y$ . But the hypotenuse is the unknown parameter here, the one you are supposed to calculate.*

S1: *Yes.*

T: *So now you have turned the problem around (...). It is just that that thought was a little backwards, maybe.*

S1: *Yes, it is quite possible.*

Following this short dialogue, the student started using MATLAB without developing an algorithmic solution or a clear strategy for solving the problem. The teacher then engaged in a discussion guiding the student step by step toward an algorithmic solution and a MATLAB function, linking the program code to the mathematical task by continuously referring to Fig. 11.2. Eventually the program was tested with  $x_1 = 0$  and  $x_2 = 1$  and output 14,142. The teacher then wanted the student to reflect on the fact that the number probably was the square root of 2, but got no response. The student thus failed to make mathematical sense of the output, realizing that the test coordinates in question would result in  $x = 1$  and  $y = 1$  as in Fig. 11.2 and thus a hypotenuse of  $\sqrt{1+1} = \sqrt{2}$ .

Referring to Fig. 11.1, an iterative step-by-step approach from mathematical task to tested program code now had been completed. The teacher then wanted the student to reflect on the process just undertaken:

T: (...). *Do you have anything to say about (...) like that afterwards?*

S1: *No, I am, I was a little bit in doubt about how to (...) First, it was the task you asked about (...) and then it was (...) and then I thought (...)  $f(x)$  is the function in  $x_2$  would be that point minus the function of that point (...) that it would be the length. But that is where I was wrong, I felt (...) Because you meant it to be here (...) and I understand that now.*

T: *Yes, that is the point, (...). That is why you have to use Pythagoras to find (...). Did you think (...) There was a hint here, wasn't there? Square root?*

S1: *Yes, yes, yes, the square root (...). I knew it was probably wrong, but I just did not quite understand what it was.*

T: *Well, because there was a clue there that you were unable to use, wasn't there. Then you realize that there is something (...)*

This excerpt shows there is little indication that the student was following a clear problem-solving strategy, which confirms that making sense of the problem and

having a mathematical understanding of the task is of crucial importance before formulating an algorithm and starting programming, which is not the case in this excerpt. A few minutes later, the teacher asked the student if there is a tendency to favor pen and paper to solve the task algorithmically before starting using MATLAB since developing an algorithm does not automatically require using the computer:

*S1: If I have it in my head, sometimes I start with MATLAB, and then I write some sort of sketch before going through it carefully. If I am not sure, I will start with paper.*

*T: Maybe the task was not quite clear?*

*S1: Yes, so far, but I had probably forgotten some of the principles there.*

*T: Principles related to MATLAB or to the mathematical assignment?*

*S1: To the mathematical problem.*

Again, the interview shows that the problem-solving process requires a good understanding of the mathematical task and computational thinking skills before programming it. Here again, the teacher reminded the student about the importance of algorithmic thinking before translating the task into MATLAB code:

*T: Now, the point of the assignment is that you should be able to translate the mathematical task into code in MATLAB. That is really the point here.*

*S1: Yes, I felt that when I understood the mathematical thing, I had no trouble putting it into MATLAB. It was simply that I had (...) forgotten a bit the length thing there. That  $f$  of that minus  $f$  of that is  $\Delta y$ , then.*

#### **11.4.2 Student 2 (S2)**

This student was an in-service mathematics teacher taking the course as a self-study. The student lived far from campus, and the interview therefore took place in the video conferencing system Zoom (<https://zoom.us/>) with only the teacher and the student present. The session was recorded, and the dialogue later transcribed. The task was sent to the student by e-mail as the interview started.

After a short dialogue with the student on the menu structure of MATLAB, which the student found unappealing, the student started studying the task. The teacher motivated by pointing out that this kind of exercise lately had become part of the exams in the programming course. Then, the teacher invited the student to reflect on the solving process:

*T: I just have to ask you ... How will you proceed to attack such a task? Will you go straight for the keyboard, and then ...? You see, you have got a small skeleton here; will you go straight for the keyboard and start programming it, or will you ...*

*S2: No, first I really try to think it through carefully, what the task is really about, here. And then I tend to make a draft as a first thing. (...) That I write down what the assignment actually asks for, and then that I make myself, in a way, a kind of design, then.*

T: A design, yes.

S2: And what is it that the program is supposed to be able to calculate here, then?  
And then, when it is plainly clear to me, I would have gone into more depth on ...  
on ... what is somehow ... how I am supposed to describe the code here. And  
here, in this case, there is a function, which is an estimate of length. And then I  
had to go in and have a look at the length. And then put it in a formula ... Oh, I  
do not know if you want me to do the task itself, kind of.

T: (...) What we are looking for, is the thought process. You say you create a draft,  
do you write code then, or draw boxes, or?

S2: Sometimes I have drawn one of these ... in a way a line, then. And then I have  
drawn a box. Or, if it involves if-statements, then I have drawn something a bit  
like this. (Shows sketch like Fig. 11.4, left, below.) If you look here. And then I  
have sort of a condition inside here, and then I have, if there is an if at the condi-  
tions, then I put this there (Shows sketch like Fig. 11.4, right, below, pointing to  
the new line.)

T: That is what we call a flow chart.

S2: Flow chart, simply.

T: Yes, that is probably something you have learned once, maybe.

S2: Yes, I learned it in this study.

This excerpt shows that the student was aware of the importance of designing an algorithmic solution before starting to program the task, in stark contrast to student 1 who struggled to understand the mathematical task and formulating an algorithm. Words like “design,” “draft,” “box,” and “function” point to algorithmic and computational thinking. Associating “if-conditions” with a flowchart also demonstrates the ability to create an abstract model from programming language construct.

Furthermore, the student makes it clear that employing CT before starting to program is a practice that has developed over time, the process being motivated by personal experience:

T: Yes, okay, yes. Very good, for it has turned out that people are very slow to  
adopt ... They often go straight on to the programming. And that works fine when

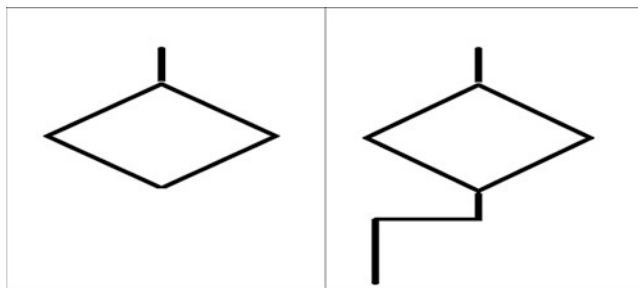


Fig. 11.4 Student's sketches

*the task is easy, but when it gets a little more complex, then you need to split things up a bit.*

*S2: Yes, I had not thought about it until I, in a way, read it in this study, but after that I have started to ... then I spend a bit of time in the process before I start typing the formula. If I have already started typing the formula, and then want to change it afterwards, then I have discovered that I do a lot of mistakes. So now I have tried to spend plenty of time on the preparatory work, so that I am one hundred per cent sure of what it is my function or program is supposed to do. And now I know it is a function, I am supposed to write, and what is it ... what is it I am actually supposed to calculate here?*

In this excerpt, the student demonstrates the ability to reflect upon own practice and followingly to abstract and articulate the essence of the process, mentioning the CT concept of dividing a task into smaller subtasks:

*T: (...)*

*S2: It is the kind of like you analyze this pretty carefully, and then you divide it, and then ... in a way ... I do not know, I work kind of structured, then, with one piece at a time.*

*T: Yes, but that is really good. It is a concept one has in programming, you make a design, then you divide it into pieces, where you can look at one piece at a time.*

Then, the student continued explaining the way of thinking algorithmically and computationally, including the programming process and the testing of the program:

*S2: And then after I have made ... made one, then ... if it ... now this task was a little bit different, then ... but ... but if I have made a program ... after I have written the entire program code ... If it typically is with for-loops and everything, then I always run through the program in my head. And write down that, okay, now,  $n$  is equal to 1, What happens to that and that value. In that way I always get to put it to the test, and then I detect possible errors.*

Referring to the step-by-step process indicated in Fig. 11.1, the student here switches back and forth between CT and programming, by alternating between a mental analysis of the mechanisms in the algorithm and testing the actual program code.

## 11.5 Discussion

This study aimed to address this research question: *How do students engage in mathematical problem-solving through computational thinking and programming activities?* The results show that the introduction of CT and programming skills into the undergraduate level presents many challenges and opportunities for students committed to improving their understanding of mathematics. The analysis of the

results shows that the participating students had two opposite experiences. They engaged differently in the mathematical problem-solving process. While the first student (S1) felt challenged by the mathematical task and the way to handle it through CT and programming, the second student (S2) made good use of CT, algorithms, and MATLAB to solve the task. The main finding of this study is that explicitly linking mathematical thinking with CT and programming is the key factor to ensure a successful implementation of the three-step iterative approach to mathematical problem-solving and enhance students' understanding of mathematical concepts.

### ***11.5.1 Summary of the Results***

For student 1 (S1), the first challenge was the lack of ability to create a mental mathematical model of the problem, hindering the student in making sense of the task, and then developing a problem-solving strategy translatable into an algorithm. This illustrates the approach presented in Sect. 2.3, requiring students to have a good mathematical understanding of the task as a basis for further development and use of computational and algorithmic thinking skills. The second challenge was related to the implementation of the algorithm using the constructs of the programming language. The student was unable to relate it to the mathematical task using an algorithm. Neither was the student able to make mathematical sense of a result output by the MATLAB program. Finally, once the teacher had guided the student step by step through the task, the student was unable to recapitulate the process. Clearly the problem-solving process in three steps outlined in the theoretical framing was challenging for the student, even after completing a programming course.

Student 2 (S2) was on the other hand very prepared to make use of CT. Student 2 had sufficient mathematical modeling capacity to make sense of the task and then to design a structured problem-solving solution translatable into an algorithm. Moreover, in contrast to student 1, student 2 demonstrated good ability to reflect, abstract, and structure and was able to use these abilities to analyze the mathematical task and to formulate an algorithm. This student was an in-service mathematics teacher and thus supposedly was trained in mathematical modeling and problem-solving prior to taking the programming course. Referring to the approach presented in Sect. 2.3, the mathematical abilities may have propagated through the steps in the model, facilitating the use of CT and programming. This is in contrast to student 1, where struggling with making sense of the mathematical task was a hindrance in going through the other steps in the approach.

### ***11.5.2 Pedagogical Implications***

Several pedagogical implications for teaching and learning CT for mathematical problem-solving can be drawn from the results. Firstly, the opposite students' experiences show that the minimum requirement for applying CT to mathematical problem-solving is a good combination of background knowledge in mathematics, algorithmic thinking skills, familiarity with the programming language constructs in question, program testing and validation, and making sense of the program output. Clearly, if students have the basic mathematical understanding required, they would be able to easily analyze mathematical tasks and break them down into small sub-tasks, designing an algorithm before moving on to programming as student 2 demonstrated in this study. These requirements and ways of problem-solving should be supported and implemented to become integral parts of university mathematics courses.

Secondly, to ensure a successful integration of CT into mathematics courses, the pedagogical setting around these courses should be well designed in terms of varied and intrinsically motivating tasks that reflect students' knowledge level. A good integration of CT into mathematics courses should also promote student autonomy and ownership. However, autonomy cannot be fully expected for novice students without good knowledge background in mathematics and some familiarities with CT and programming language constructs. Clearly, CT is a difficult matter for novice students, because it is more a way of thinking than a computational skill to acquire.

Thirdly, as this study shows and following the second implication, the role of the teacher as a facilitator of learning is still crucial to assist novice students in mathematical problem-solving by means of CT and programming. The role of a knowledgeable teacher is to foster CT and connections between mathematical thinking, programming, and algorithmic thinking.

Finally, following the second and third implication, a pedagogical setting that promotes a learning environment capable of motivating students should be created around mathematical problem-solving, CT, and programming practices. In other words, to alleviate the difficulty of thinking computationally, the three-step approach presented in this study needs to be combined with a pedagogically sound model, which emphasizes two elements: The teacher as facilitator of learning, and peer collaboration to allow the more competent students help those facing difficulties in accomplishing the computational tasks (Kuo & Hsu, 2020). Research on peer collaboration may contribute to explore collaborative learning occurring in the process of learning CT and programming.

### ***11.5.3 Limitations of the Study***

While the results from the opposite students' experiences and insights of the present study contribute to offer a better understanding on the interactions between CT, mathematical thinking, and programming, the study is limited to two students and can therefore not be generalized to all students enrolled in the course. The study is also limited to a very specific mathematical task (Pythagoras' theorem) involving undergraduate students with no prior experience in CT and programming. A final limitation that may have influenced the study is the method being used to gather data, that is, the semi-structured interview with the students once they have finished the course, one performed online.

## **11.6 Conclusions and Future Research**

Although the participating students may not be representative for the average student enrolled in the course and the task was specific, two preliminary conclusions can be drawn from the study. Firstly, the connections between mathematics, CT, and programming languages are quite complex in line with the research literature (Li et al., 2020a, 2020b) and need to be clearly articulated in authentic mathematics educational settings. Secondly, engaging in mathematical problem-solving through CT and programming requires good background in mathematics, algorithmic thinking, and familiarity with the programming language constructs.

The study is still a work in progress. The authors will continue working with undergraduate students and programming courses with applications in mathematics using interventions involving a wide range of participants and whole classes. Future work aims at investigating the following issues.

Firstly, future research will analyze the application of CT in varied, more differentiated, and ill-defined mathematical tasks to deepen the understanding on how CT skills are deployed and applied to solve these kinds of mathematical tasks.

Secondly, future research will focus on more elaborated and in-depth analysis of students' experiences with mixed methods to ensure more reliability and validity of the results. Work will include more participants and a mix of quantitative and qualitative methods to assess the impact of CT on students' learning and uncover what benefits and obstacles different students actually experience in authentic educational settings while thinking computationally to solve mathematical problems, in line with the research literature (Broley et al., 2018).

Thirdly, in terms of the theoretical framework being used in this study, future research work will explore the three-step iterative approach to mathematical problem-solving in more details and depth, in particular the interactions between mathematical thinking, CT, and programming to highlight their communalities and potential differences and suggest changes to the proposed approach. Future research will also look at alternative theoretical frameworks and learning theories used in



mathematics education such as Vygotsky's sociocultural theory, Trouche's instrumental approach, Brousseau's theory of didactical situations, or Chevallard's anthropology theory of didactics, but also constructionistic approaches to examine carefully whether these frameworks and theories can be combined and coordinated with the approach presented in this study.

Fourthly, future research should consider usability issues of programming languages since technicalities and familiarity with the language are prerequisites for mathematical problem-solving. Ease of use and understanding of language constructs, interactions, and program feedback of the programming language should be considered in future implementations since these are useful to foster successful implementations of algorithmic solutions to mathematical problem-solving.

Finally, future research will conduct an in-depth examination of how teachers apply CT and programming. This would allow researchers to further probe what teachers view, understand, and interpret as CT and explore how mathematical problem-solving relates to CT and programming from the teacher's point of view.

## References

- Artigue, M., et al. (2009). Connecting and integrating theoretical frames: The TELMA contribution. *International Journal of Computers for Mathematical Learning*, 14, 217–240.
- Bokhove, K., & Drijvers, P. (2010). Digital tools for algebra education: Criteria and evaluation. *International Journal of Mathematics Learning*, 15, 45–62.
- Brolley, L., Caron, F., & Saint-Aubin, Y. (2018). Levels of programming in mathematical research and university mathematics education. *The International Journal of Research in Undergraduate Mathematics Education*, 4, 38–55.
- Buteau, B., Muller, E., Marshall, N., Sacristán, A. I., & Mgombelo, J. (2016). Undergraduate mathematics students appropriating programming as a tool for modelling, simulation, and visualization: A case study. *Digital Experiences in Mathematics Education*, 2, 142–166.
- Buteau, C., Muller, E., Mgombelo, J., Sacristán, A. S., & Dreise, K. (2020). Instrumental genesis stages of programming for mathematical work. *Digital Experiences in Mathematics Education*, 6, 367–390.
- Buteau, C., et al. (2018). Computational thinking in university mathematics education: A theoretical framework. In A. Weinberg et al. (Eds.), *Proceedings of the 21st annual conference on research in undergraduate mathematics education* (pp. 1171–1179). RUME.
- DeJarnette, A. (2019). Students' challenges with symbols and diagrams when using a programming environment in mathematics. *Digital Experiences in Mathematics Education*, 5(1), 36–58.
- Filho, P., & Mercat, C. (2018). Teaching computational thinking in classroom environments: A case for unplugged scenario. In V. Gitirana et al. (Eds.), *Proceedings of the re(s)ources 2018 international conference*. ENS de Lyon.
- Gueudet, G., et al. (2020). *Programming as an artefact: What do we learn about university students' activity? INDRUM 2020*. Université de Carthage, Université de Montpellier, Cyberspace (virtually from Bizerte), Tunisia. Retrieved from <https://hal.archives-ouvertes.fr/hal-03113851/document>
- Hadjerrouit, S., & Gautestad, H. H. (2019). Evaluating the usefulness of the visualization tool SimReal+ for learning mathematics: A case study at the undergraduate level. In D. Sampson et al. (Eds.), *Learning technologies for transforming large-scale teaching, learning, and assessment* (pp. 71–89). Springer Nature Switzerland AG.

- Kotsopoulos, D., Floyd, L., Nelson, V., Makosz, S., & Senger, N. (2019). Mathematical or computational thinking? An early years' perspective. In K. M. Robinson, H. P. Osana, & D. Kotsopoulos (Eds.), *Mathematical learning and cognition in infancy and early childhood: Integrating interdisciplinary research into practice*. Springer.
- Kuo, W.-C., & Hsu, T.-C. (2020). Learning computational thinking without a computer: How computational participation happens in a computational thinking board game. *The Asia-Pacific Education Researcher*, 29(1), 67–83.
- Lee, I., & Malyn-Smith, J. (2020). Computational thinking integration patterns along the framework defining computational thinking from a disciplinary perspective. *Journal of Science Education and Technology*, 29, 9–18. <https://doi.org/10.1007/s10956-019-09802-x>
- Li, Y., et al. (2020a). On computational thinking and STEM education. *Journal for STEM Education Research*, 3, 147–166.
- Li, Y., et al. (2020b). Computational thinking is more about thinking than computing. *Journal for STEM Education Research*, 2020(3), 1–18.
- Lie, J., Hauge, I. O., & Meaney, T. J. (2017). Computer programming in the lower secondary classroom: Mathematics learning. *Italian Journal of Educational Technology*, 25(2), 27–35.
- Malyn-Smith, J., & Angeli, C. (2020). Computational thinking. In A. Tatnall (Ed.), *Encyclopedia of education and information technologies* (pp. 333–340). Springer Nature Switzerland AG.
- Misfeldt, M., & Ejsing-Duun, S. (2015). Learning mathematics through programming: An instrumental approach to potentials and pitfalls. In K. Krainer & N. Vondrová (Eds.), *Proceedings of CERME9* (pp. 2524–2530). Prague, Czech Republic.
- Papert, S., & Harel, I. (1991). *Constructionism*. Ablex Publishing.
- Patton, M. Q. (2002). *Qualitative research and evaluation methods*. Sage Publications.
- Pérez-Marín, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children? *Computers in Human Behavior*, 105, 105849.
- Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education*, 14, 1–15.
- Santos, S. C., Tedesco, P. A., Borba, M., & Brito, M. (2020). Innovative approaches in teaching programming: A systematic literature review. In *Proceedings of the 12th International Conference On Computer Supported Education (CSEDU 2020)* (Vol. 1, pp. 205–2014). Prague, Czech Republic.
- Shodiev, H. (2015). Computational thinking and simulation in teaching science and mathematics. In M. G. Cojocaru et al. (Eds.), *Interdisciplinary topics in applied mathematics* (Springer proceedings in mathematics and statistics) (Vol. 117, pp. 405–410).
- Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142–158.
- Topalli, D., & Cagiltay, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with scratch. *Computers and Education*, 120, 64–74.
- Weintrop, D., et al. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25, 127–147.
- Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A*, 366(1881), 3717–3725.
- Wing, J. M. (2014). Computational thinking benefits society. *Social issues in computing*, 40th Anniversary Blog, University of Toronto. Retrieved from <http://socialissues.cs.toronto.edu/index.html%3Fp=279.html>