Facilitating exploratory talk through mathematical programming problems

MORTEN MUNTHE

With several Nordic countries implementing programming into their curricula, there is a need for research into the combination of mathematical learning and programming. With tasks being the main "thing to do" in the mathematics classroom and exploratory talk being closely linked to learning this article investigates what contributes to and what hinders exploratory talk when working on mathematical programming problems. The data collection comprises of recordings of students working on tasks in a mathematics classroom in Norway. The findings indicate that programming can facilitate mathematical talk in the classroom, that programming is best implemented to facilitate in-depth learning, and that adversities are present in both instances.

The implementation of programming, often in combination with algorithms, into the curricula in many European countries (Bocconi et al., 2022) and the lack of research into how to facilitate the implementation of programming, especially in upper secondary school (see Lv et al., 2023) is a challenge. In Sweden and Norway programming have been implemented into the mathematics curriculum (Bocconi, 2018; Skolverket, 2019; Utdanningsdirektoratet, 2020). Previous research has shown that programming improves students' logical thinking (Park et al., 2015) and their understanding of mathematical processes (Calao et al., 2015), which can lead to more joyful learning processes (Djurdjevic-Pahl et al., 2017) and strengthen students' self-confidence (Shim et al., 2016). Combining student interaction with task design, where students rely on each other to generate, challenge, refine and pursue new ideas, has been shown to be beneficial (Francisco & Maher, 2005). Small group collaborative learning in school mathematics can bring about more equal academic success amongst all students compared to traditional methods

Morten Munthe
Norwegian University of Life Sciences

Munthe, M. (2024). Facilitating exploratory talk through mathematical programming problems. *Nordic Studies in Mathematics Education*, 29 (1), 61–82.

of teaching (Davidson & Kroll, 1991; DePree, 1998; Slavin, 1990; Urion & Davidson, 1992). Exploratory talk, where students engage critically but constructively with each other in small groups (Mercer, 2005; Mercer & Littleton, 2007), is used here as it has been shown to stimulate subject learning and reasoning skills (Knight & Mercer, 2015; Mercer et al., 2004; Mercer & Sams, 2006). The aim is to investigate what hinders and facilitates exploratory talk in students in upper secondary school when working on mathematical programming problems.

A mathematical programming problem (MPP) is a series of tasks designed to combine mathematics with programming to resolve a problem (Munthe, 2022a, 2022b). Through an analysis of the interaction between students working in groups on MPPs, the article investigates elements contributing to and hindering mathematical exploratory talk.

Background for and the design of MPPs

The design of the MPPs is presented in-depth in other works (see Munthe, 2022a, 2022b), but as it is important, a short description is provided. The following section presents research in learning programming, then applying the theory of didactical situations (TDS) to build the foundation for combining mathematical learning and programming in the design. Finally, the design of the MPPs is presented followed by challenges with implementing programming in the classroom.

Theory

Research into computer education indicates that learning to program is difficult as "students exhibit various misconceptions and other difficulties in syntactic knowledge, conceptual knowledge, and strategic knowledge" (Qian & Lehman, 2017, p. 17). Ko et al. (2004) use the term *barriers* to differentiate between six different types of adversities students encounter when learning to build a program. These barriers are a combination of syntax errors, structural errors, logical challenges, error handling issues and problems related to the number of commands available (Ko et al., 2004). When combined with the difficulty of creating tasks using digital technologies (Joubert, 2007; Laborde & Sträßer, 2010), text-based programming is expected to present difficulties when introduced and implemented into the mathematics classroom.

According to the theory of didactical situations (TDS), mathematical learning is more likely to take place when students are committed to a solving a problem (Brousseau 1997) where knowledge is defined as a pro-perty of a system consisting of a subject and a milieu. As students

work on the problems, they interact with the milieu creating an *adidactical situation*, where students show initiative and responsibility for the outcome of the learning process. Designing problems that facilitate students adapting their strategies to obtain the desired knowledge is challenging. and obstacles in TDS are a central part of this adaptation. An *epistemological obstacle* constitutes a form of knowledge that has been relevant and successful, often in school contexts, but becomes false or insufficient at a particular moment in time. If students overcome such an obstacle through the adaptation of their strategies, the desired knowledge can be obtained. O*ntogenic obstacles* relate to the limitations of students and a lack of required prior learning, and *didactical obstacles* relate to the presentation of the subject, "the result of narrow or faulty instruction" (Harel & Sowder, 2005, p. 34). If students encounter and overcome obstacles when working through the given problems, adaptation may take place. Similarly, Stein et al. (1996, p. 426) state that "tasks used in mathematics classrooms highly influence the kinds of thinking processes in which students engage, which, in turn, influences student learning outcomes." The aim of MPPs is to create adidactical situations (Brousseau, 1997), facilitating mathematical exploratory talk during which students reconsider their strategies, develop new pathways, discuss with their peers, conjecture and experiment, all related to the intended learning process (Leung & Baccaglini-Frank, 2016). To facilitate adidactical situations, an MPP is structured using seven steps, see figure 1.

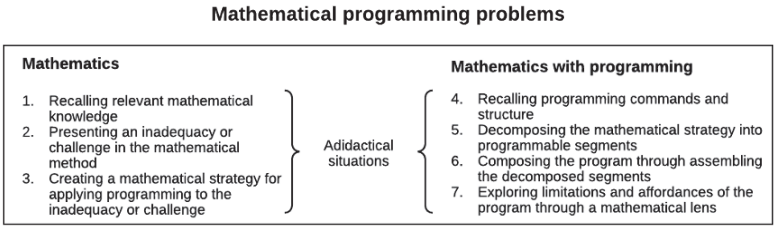

Figure 1. *The seven steps for designing MPPs*

Design

The structure of the design is the product of a three-year iterative process of implementing MPPs in an upper-secondary school mathematics class (see Munthe, 2022b). As an example, I will use the building of a program to calculate the zero-point of a function using the bisectional method[1]. Step 1 consists of asking the students to find the zero-points for several different types of functions, exemplifying the different methods required for each function type. Step 2 problematizes finding the zero-point if the

given function, such as $f(x) = \ln(x+2) - x$, does not conform to any set method. In step 3, one student takes the role of a "program" and is given a set of graphs where the function is unknown, but the zero-points are provided. The rest of the group is tasked with finding the zero-points by giving the "program" x-coordinates, and the "program" can only respond with the accompanying function values. This allows several strategies for finding the zero-point to be tested and evaluated. In step 4, tasks allowing students to recollect necessary programming structures and commands are given. Step 5 consists of transforming the method discovered in step 3 into a coding structure, and step 6 consists of building the program. Both steps 5 and 6 contain tasks to assist with aspects of code building. The final step asks the students to evaluate the program, investigate whether it works for all functions and provide explanations of why it does not work for some functions if that is the case. An in-depth description of the design is presented in Munthe (2022b).

Tasks using programming need to avoid overloading students with complexity and programming syntax (Ko et al., 2004), and combined with mathematics this becomes even more important. An investigation of mathematical textbooks in Sweden shows that tasks are often in the form of "follow a procedure", and not using programming facilitating mathematical exploration (Bråting & Kilhamn, 2022), which is central to the MPPs. Research show that complex problem-solving without sufficient support structures can result in an unproductive cognitive process (Kirschner et al., 2006; Reiser, 2004). In task design, insufficient support structures often result in either didactical or ontogenic obstacles (Brousseau, 1997). Based on this research, this article collectively refers to *adversity* when the students encounter an obstacle, but does not view adversity as a negative as it can lead to a productive struggle, where making connections between mathematical constructions visible to the students (Hiebert & Grouws, 2007).

Theory

Here, mathematical learning is linked to exploratory talk and the interaction between students. This is followed by the reviewing literature regarding small-group interactions and the relation to programming. The interactions are then presented in more detail building the foundation for the coding scheme.

Learning in mathematics can be seen as "the construction of a web of connections – between classes of problems, mathematical objects and relationships, real entities and personal situation-specific experiences" (Noss & Hoyles, 1996, p. 105). The linking of concepts is also well

established in the adidactical situations in TDS, and part of the design of the MPPs Facilitating students to explain their reasoning and providing warrants for their arguments while recognizing them as contributors are part of the exploratory talk that takes place between the students (Choi & Walters, 2018; Mercer, 2005). Exploratory talk is defined as a dialogical feature where

> partners engage critically but constructively with each other's ideas. Statements and suggestions are offered for joint consideration. These may be challenged and counter-challenged, but challenges are justified, and alternative hypotheses are offered. Partners all actively participate, and opinions are sought and considered before decisions are jointly made. (Mercer, 2005, p. 9)

This definition coincides with the aim of the MPPs, where student discussion is central. The role of mathematical talk and discussion in learning (Kazemi & Stipek, 2009; Resnick et al., 2017) and in the effective teaching and development of mathematics (Cobb et al., 1997; Sfard, 2000) is well documented, and the foundation for this article and the MPPs is that exploratory talk can promote mathematical learning.

Small group learning in mathematics, especially in conjunction with technology, has demonstrated notable advantages (Berry & Sahlberg, 2006). Lou et al. (2001, p. 449) affirm that when technology is integrated with exploration, small group learning yields significantly greater benefits compared to individual learning. In this study, students were organized into groups to facilitate exploratory talk. This process encompasses distinct stages, starting with an initializer, which may manifest as a claim, a suggestion, or the presentation of a problem. Toulmin (1969) defines a claim as an assertion, distinguishing it from hypothetical or frivolous statements. Conversely, a suggestion often takes the form of stating a hypothesis (Pedaste et al., 2015), demanding varying degrees of attention. While both a claim and a suggestion are propositions, they diverge in the certainty of their assertions. When confronted with a problem, students frequently draw upon prior knowledge to initiate the talk. This collective reflection not only forms the foundation for problem-solving but also underpins the creation of new knowledge (Tabach & Schwarz, 2018; Yackel & Cobb, 1996).

The usage of short exercises promoting curiosity and eliciting prior knowledge is essential to facilitate engagement, and the exploration that follows is the result of a combination of promoting curiosity and accessing previous knowledge to ensure that the students are prepared for the learning outcomes of the current problem (Bybee et al., 2006). The ensuing exploration, driven by "what if" inquiries can help the

students locate new mathematical ideas (Alrø & Skovsmose, 2004). Curiosity, when integrated with presenting a challenge, can facilitate a process of engagement (Pedaste et al., 2015). Exploration is a sequence of activities initiated by the problem and undertaken by the students, using prior knowledge to generate new ideas, explore possibilities and undertake an investigation (Bybee et al., 2006). Both engagement and exploration are observable through the lens of exploratory talk (Mercer, 2005).

Student engagement and exploration in mathematics education has been a focal point in the literature. This is particularly evident in the emphasis on explanation as a means for students to manifest their grasp of the concepts involved (Bybee et al., 2006). By prompting students with "why" questions, they are not only encouraged to delve into deeper understanding but also to unearth novel mathematical ideas (Alrø & Skovsmose, 2004). It is crucial to recognize that elucidation serves students progressing towards the intended learning objectives of a task (Bybee et al., 2006). This notion aligns with Bybee et al.'s (2006) sequential instructional model (Engage, Explore, Explain, Elaborate and Evaluate), which has been adapted to fortify the framework under consideration. Within the realm of exploratory talk, it becomes evident that explanation and evaluation are integral components. This entails synthesizing acquired data to forge new knowledge and engaging in discussions that challenge or counterchallenge proposed ideas (Pedaste et al., 2015). Mercer (2005) advocates for making the reasoning process of the group explicit, thereby necessitating individuals to articulate their thoughts and subject them to scrutiny from their peers (Alrø & Skovsmose, 2004). These reflective activities have been shown to be instrumental in facilitating learning (Pedaste et al., 2015). Moreover, the outcome of the explanation and evaluation phases in exploratory talk can be transformative, moving the mathematical talk in new directions (Alrø & Skovsmose, 2004; Gellert, 2014). This can culminate in a joint decision or agreement regarding the clarification of a given problem or a specific result. However, it is imperative that such consensus be substantiated with a comprehensive account of how the group arrived at its conclusion (Drageset, 2014). In essence, agreement is like closing, signifying the conclusion of the ongoing discussion pertaining to a given problem (Gellert, 2014).

The exploratory talk the students engage in can be facilitated by the teacher, who can engage in a series of dialogic acts with the students such as challenging established knowledge (Alrø & Skovsmose, 2004). The MPPs were created to facilitate exploratory talk with minimal assistance from the teacher creating adidactical situations where challenging the established knowledge comes from the epistemological obstacles that can occur.

Method

The MPPs investigated were implemented in a classroom as part of an advanced mathematics course for students ($n = 28$) aged 17 in the second to last year of secondary school. Three groups were chosen based on a combination of different genders and grades, and most importantly, on the individual students' ability to sufficiently convey their ideas vocally during a lesson. The three groups consisted of four males, two females, and one male and one female, all of whom had collaborated well throughout the year. The size of the groups and their composition was based on the students' preferences with the approval of the teacher. The four males were working well together and as such they were allowed to remain in a group of four. The group size was not seen to have a significant effect on the discussion. The students were told to discuss the problems within their groups, allowing them to work on and discuss any adversities before receiving help from the teacher. The teacher was used as a "last resort", and as such the teacher's role is not the focus in this work.

The data collection consisted of recording each student's computer screen together with their voice. The screen and voice recordings for each member of the group were digitally combined to form one video file with two to four screens and an audio where all the students' voice recordings were combined. This resulted in several advantages regarding the transcription. First, viewing all the screens together provided a clear indication of where each student was in the process of building the program. Second, the combination of voice recordings made each student's contribution clear, allowing for an accurate transcription.

The analytical framework used consists of three layers of coding. The *segment* category divides the lesson into smaller sections, in which the students talk about one problem or challenge encountered. A segment is defined as *one problem or one part of a problem, initialized by a problem separate from the previous one, and ending with either an agreement regarding the given problem or the change to a new problem*.

The *subject* category refers to the topic of the interaction. The first type is mathematical, in which the students talk about a mathematical object, such as the derivative or the number of solutions. The second is programming, in which the students talk about programming, for instance, how different commands work, the structure of a code segment or what caused a programming error. The final group is a combination of the two, including talks of how to combine programming with a mathematical objective, for instance, "How do we avoid the program dividing by zero?" or "I want the program to do <a mathematical procedure>. How do I accomplish that?"

The *interaction* category consists of exploratory talk and adversity. Exploratory talk consists of initiation, explanation, exploration and agreement. Initiation allows for the compartmentalization of the transcript into segments. Explanation, as defined by Mercer (2005), is a crucial aspect of students' discourse, where they validate their approaches, critique suggestions, and justify solutions. It also involves recalling prior knowledge, particularly in handling mathematical and combination challenges (Pirie & Martin, 2000).Exploration is pivotal in mathematical programming, aiding students in troubleshooting errors through collaborative discussions (Benton et al., 2017). This entails applying mathematical principles and programming techniques to test different problem-solving strategies. Such exploration may involve code testing, mathematical evaluations, and probing the program's capabilities. Through exploration, the students investigate the limits and possibilities of the program, leading to mathematical questions, such as "why is the derivative of $\log(x)$ not showing for negative values?" and the mathematical evaluation of the results, such as "why is the result of the calculation not as expected?" Agreement signifies collective consensus on specific answers or code snippets, marking a significant milestone in problem-solving. Disputed statements do not qualify as agreements, which can pertain to mathematical, code-related, or hybrid aspects. Agreements can either conclude a problem or instigate further proposals, evaluations, or explorations. While not every segment reaches an immediate agreement, explanation and exploration often drive the process in new directions. If the students are unable to reach an agreement through their own interaction, they usually result to asking the teacher, who, through discussion with the students, assists them in reaching an agreement.

Adversity is defined here as situations where the students display uncertainty regarding how to proceed with the MPP they are working on. From a mathematical perspective, these are the instances where the students encounter obstacles (Brousseau, 1997), and from a programming viewpoint, barriers (Ko et al., 2004). These can be observed as frustration, and commonly occur when executing the program and receiving an error message, but also appear in terms of difficulties distinguishing between commands, sequencing challenges, uncertainty regarding how to proceed and a general sense of a negative premonition. This feedback can lead to discouraging self-relevant interpretations (Fyfe & Brown, 2020), which make the students give up and link their frustrations to mathematics. Adversity by itself does not distinguish between different "types" of obstacles. Only when viewing adversities in relation to exploratory talk does the difference between an epistemological and another type of obstacle become clear.

Table 1. *Coding scheme for the analysis of the transcripts*

Code for interaction	Description
Exploratory talk	Engagement within the group consisting of ideas, suggestions, challenges and justifications
Initiation	Start of a segment
Explanation	Explaining and arguing for a solution Presenting criticism and suggestions Validating a solution
Exploration	Testing a code Running the program
Agreement	Reaching common ground
Adversity	The group displays uncertainty regarding how to proceed
Positive	Leading to or facilitating exploratory talk (epistemological obstacles)
Negative	Leading to frustration and a "this is not going to work"-mentality (didactical and/or ontogenic obstacles)

Within each segment, the transcription was coded for both subject and interaction, enabling an analysis of the relationship between exploratory talk and adversity and their relation to mathematics, programming, or a combination of the two. When a student engages in a *mathematical explanation*, they use their mathematical knowledge and vocabulary to present their reasoning. In a *combination explanation*, they include either the code itself or the output from the program to assist their reasoning, and in a *programming explanation*, they use the different commands and programming structures for their reasoning and do not apply mathematical vocabulary. To ensure intercoder reliability, a transcript containing 58 utterances was coded by two additional researchers. Of the 58 utte-rances, there was a deviation from one of the coders on 7 utterances, and a deviation from both coders on 1 utterance. For subject, all instances of deviation came from one coder selecting combination, and the other selecting either mathematics or programming. For interaction, the deviances occurred when a statement from a student contained both exploration and adversity, and one was favored over the other.

The first MPP concerns the bisectional method (see the example described above in the design of the MPP). To illustrate a problem containing an adversity that intends to facilitate exploratory talk from the MPP concerning the bisectional method, see figure 2.

The second MPP asks the students to build a program implementing Newton's method for approximating the zero-point[2]. The MPP starts with a recollection of the derivative and its application within function analysis. The MPP facilitates the students in building visual and

Imagine a machine that contains and unknown linear function and will give you a function value when you input a numerical x-value. All non-numerical inputs will yield an error.

How can you systematically use this machine to find the zero-point for the unknown function to the right? What if the function is non-linear?

The machine can be as accurate as you would like concerning the function value. Discuss possible strategies with your group.

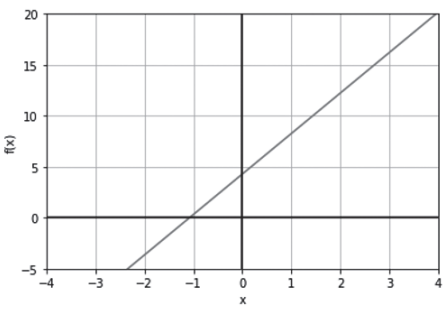

Figure 2. O*ne problem from the first MPP covering the bisectional method*

Does Newton's method work for all functions?

a) Draw a function that crosses the x-axis in a coordinate system. Choose an x-value close to the zero-point and draw a tangent line to find a new value for the zero-point. Continue this process a few times. Do you get any closer to the zero-point with each new tangent line?

b) Repeat a) a couple of times with different types of functions. Are there certain functions or types of zero-points for which the method does not work? If so, why? Discuss with your group.

Figure 3. O*ne problem from the second MPP covering Newton's method*

Does Newton's method work for all functions?

a) Test the program for several different functions, e.g. different polynomial functions, exponential functions, logarithmic functions.

b) What characterizes the functions for which the program does not work? Discuss with your group.

Figure 4. O*ne problem from the second MPP covering Newton's method*

algorithmic proof for the method, followed by implementing the method into a program. When the students have built the mathematical argument for Newton's method, they are asked to evaluate it (see figures 2 and 3). When the students have finished building their programs, they are asked to complete another similar problem (see figure 4).

Results

Bisectional method

Prior to this lesson, the students completed programming tasks involving building programs covering graph plotting, calculating areas and circumferences of geometric shapes, and solving linear and quadratic equations.

This MPP consisted of the students building a program to find the zero-point for a function that crosses the x-axis using the bisectional method. The students were familiar with different methods for finding the zero-point of functions but had not previously used a numerical method to find the zero-point.

The extract in table 2 illustrates the development of a strategy by asking one student from each group to simulate the machine (see step 3 of the example described above in the design of the MPP). The machine-simulation student is A and the zero-searching students are B and C. The group have read the problem, and A has agreed to simulate the machine. The transcript starts with student A expressing initial uncertainty about what to do (101). This is quickly resolved by the other members of the group through explanation and an example (102–104). After a short pause, where student A studies the graphs, the students start enquiring about the function values for a couple of x-values (105–108). Drawing on the information gathered, a mathematical evaluation and suggestion is made (109) and swiftly agreed upon by the rest of the group (110). This agreement also encompasses verification since the machine-simulating student confirms the suggestion. Finally, student C develops the idea further with both a suggestion and an explanation for the proposal. The segment contains combination adversity, and combination and mathematics exploratory talk.

Table 2. *Transcript of students working on the MPP covering the bisectional method*

	Student	Transcript	Code
101	A	What am I supposed to do now? [referring to the task in figure 2]	Combination – Adversity
102	B	You are now python [the programming language]	Combination – Explanation
103	C	You are going to say ...	
104	B	For example, what is y when x is ...?	
<short pause while A looks at the graphs given>			
105	B	What is y when x equals zero?	Combination – Exploration
106	A	Then it [the y-value] is minus one	
107	B	But then we take x equals one, right?	
108	A	x equals one ... that is ehm ... yes, that is approximately, approximately zero point eight	
109	B	ahh, but then we have at least got a zero-point in between there	Mathematics – Explanation
110	A and C	that is so true	Mathematics – Agreement
111	C	We can actually make it even more accurate and say that since we know there is a zero-point between these numbers ...	Combination – Explanation

The MPP elicited both success and frustration: success in terms of progress towards building a better method for guessing, and frustration in that the function could have more than one zero-point, both contributing to an adidactical situation. There were some adversities in the implementation of the method they uncovered in the first part, but they were mostly able to build the program. There were several instances where the students explored the program by inputting different types of functions. The interaction between the students consisted of discussing the code and the mathematical verification of the result based on the output of the program. The overall impression obtained from the lesson was that the students understood the bisectional method.

The students' work on the bisectional method is visualized in the graph (figure 5), where almost every single transcript segment related to mathematics. Since the program did not demand any overly complicated code, the students were observed as having fewer problems with the coding part of the MPP, hence the low incidents of programming dialogue. Additionally, since the MPP was focused on revisiting previous knowledge and games at the start, there were few instances of adversity. Each exploratory talk, especially regarding mathematics and combination, involved several steps and outnumbered the incidents of adversities in the lesson. A typical occurrence to check the validity of the program involved finding a suitable function and solving it mathematically by hand, followed by inputting the function and an initial guess into the program. When the program produced a result, they compared the output to the exact answer and evaluated whether the numerical method was close enough. The few instances of adversity consisted of designed tasks within the MPP, such as, "what is a quick way to check whether numbers

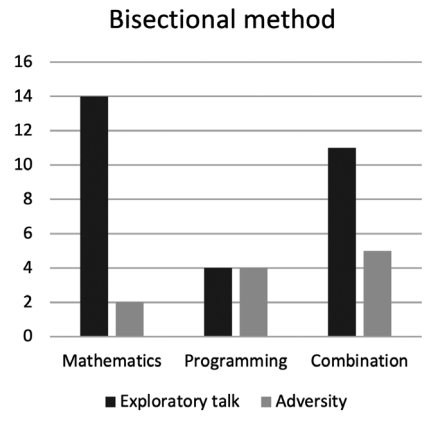

Figure 5. *Graph showing instances of the categories in the transcript*

have the same or different signs?"; this effectively created adversity in the design to be solved through exploratory talk.

Newton's method

In between the two classes, the students completed a lesson covering the numerical derivative. The second MPP introduced the students to the combination of both learning a new formula (Newton's method) and implementing the formula into a program. The students had never seen a formula that included both a function and its derivative. Combining all these elements is difficult, but the students had now used programming throughout the school year and were expected to be more confident as a result. The MPP started by explaining the origin of Newton's method and why it, in most cases, will yield a good x-value for the zero-point after only a few iterations.

In this transcript (see table 3), the students are working on a code that calls on a function and have just executed the program and received an error.

After receiving a syntax error, L expresses frustration and cannot initially recognize why the program is at fault (201–203). M recognizes that a syntax error often occurs from a misuse of parentheses (204) before they together locate the line of code containing the error (205–206). By applying an evaluation of both the code and the mathematics, a correction is then made (207) before they run the program again. After the program executes and displays the graph, they agree on the correction

Table 3. *Transcript of students facing an adversity when working on an MPP*

	Student	Transcript	Code
201	L	but yes, I have something wrong here	Programming – Adversity
202	M	return d y <reading the code>	
203	L	I do not understand what is wrong, but there is something wrong. It says invalid syntax	
204	M	Oh, then you have a wrong parenthesis	Programming – Explanation
205	L	... in this one?	
206	M	In that one <points to the code on the screen>	
207	L	Oh yes, it is missing a bracket here ... there <moves the cursor to point at it and inserts a bracket>	Combination – Explanation
<runs the program>			
208	L	Now it works	Combination – Agreement
209	M	Yeah	

made (208–209). The extract contains programming adversity and programming and combination exploratory talk.

The students' conversations were dominated by adversity relating to how to construct the program rather than mathematical exploratory talk of how the formula worked. At the end of the lesson, the impression was that the students had constructed a program but had limited knowledge of how the mathematics behind the code worked.

The graph (figure 6) illustrates the prominence of both adversity and exploratory talk concerning programming and the combination of programming and mathematics. It is important to highlight that the combination dialogue, where there is mathematical talk, consisted primarily of the implementation of the formula into the code rather than actual discussion of Newton's formula. The MPP contained exploratory problems, which in the previous lesson had aided the students in discussing the mathematical formula applied, but here, the students ended the discussion quickly to instrumentally program Newton's formula. As a result, the adversities started out as combination adversity and deteriorated into relating only to programming. Most of the programming adversities resulted from errors, which was followed by the students trying to locate the error, often through combination knowledge, before either succeeding or failing at correcting the error. There were instances where the students completed several iterations of receiving an error, making a correction, running the program and receiving a new or identical error. Every single segment contained programming in one form or another, and the mathematics, although present in the combination dialogue, was lacking.

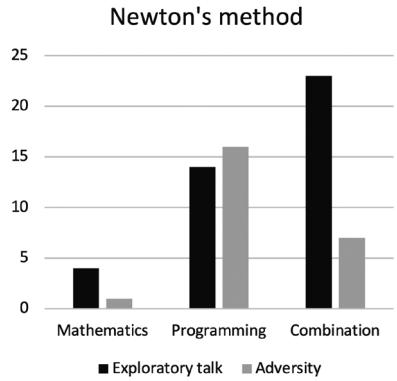

Figure 6. *Graph showing instances of the categories in the transcript*

Discussion

Exploratory talk is consistently represented related to either mathematics or the combination of programming and mathematics, indicating that it is possible to implement programming into mathematics classrooms and facilitate exploratory talk. In both lessons presented, exploratory talk is present in almost every communication between students, and each lesson reveals a significant number of interactions between the students.

The amount of exploratory talk versus adversity is different in the two lessons. The bisectional method lesson contains several instances of adversity, but with the significant amount of exploratory talk that takes place, the adversities are resolved and become an epistemological obstacle (Brousseau, 1997). The low representation of programming issues is likely due to the familiarity with the mathematical foundation and the low complexity of the programming required. This lesson indicates how it is possible to introduce a new subject using programming when students have a strong foundation. When students are learning to apply programming in mathematics, it is beneficial that they are familiar with the mathematical concepts involved so that they are not met with two complexities at the same time (Kirschner et al., 2018; Ko et al., 2004; Reiser, 2004). Applying programming to a well-known mathematical method alleviates one part of this two-sided challenge. In the Newton's method lesson, adversity features more prominently. The expectation was that there would be some discussion regarding the mathematical properties of Newton's formula and its implementation into the program, but only the latter occurred. The lack of mathematical exploratory talk suggests that the MPP did not manage to create a link between students' previous knowledge of the derivative, the tangent of the derivative and the zero-point of the function. This lesson indicates that when using programming to introduce a new method in mathematics, care must be taken that the programming does not overshadow the mathematical content of the lesson. The difference is that every mathematical step of the bisectional method was already known to the students. In Newton's method, while they had been taught the derivative and application of the tangent, using the derivative to find a zero-point was new. This created an increased amount of adversity for the students, leading to less mathematical exploratory talk.

The bisectional method lesson facilitated exploratory talk indicating that students are better prepared to learn from building a program when there is less adversity and when more support structures are in place (Drijvers, 2012; Kirschner et al., 2018; Reiser, 2004). This is not to say that situations do not exist where both learning a mathematical concept and

build a program can occur, but accomplishing both during one lesson is challenging. Designing problems that apply known mathematical concepts together with programming yields opportunities to facilitate a deeper understanding of the underlying mathematics for the learner.

The design idea was that programming can be a tool for learning mathematics. In the bisectional method MPP, the tasks facilitated the students in uncovering the bisectional method themselves before having to create a program using this method, generating a good support system (Drijvers, 2012), contributing to building an adidactical situation (Brousseau, 1997) and facilitating exploratory talk (Mercer, 2005). Numerous mathematical exploratory talks occurred, probably due to the introduction and application of well-known methods, together with the fact that the program did not contain any procedures that the students were unfamiliar with. This allowed the students to focus on understanding the mathematics behind the method instead of meandering through programming code, in which didactical and ontological obstacles (Brousseau, 1997) are more likely. In the Newton's method MPP, the balance was skewed towards programming, and the exploration suffered because the students focused extensively on getting the program to work, rather than understanding the mathematical principles behind the method. The investigation shows that the creation of a program together with a relatively new mathematical method created ontogenic and didactical obstacles for the students, resulting in them focusing more on the instrumental creation of the program.

The main adversity hindering exploratory talk that students encounter when working with programming is the building of the program through writing code segments. An argument can be made that with time students will have more programming knowledge and MPPs such as Newton's method could work, but until that time it is worthwhile to consider the role and design of the task. The hindrance is twofold, where the first obstacle is the coding itself. The implementation of digital tools in the classroom has been researched previously (Drijvers, 2012), but most digital tools used in schools are designed to assist the students using them. They often make use of methods, such as graphical interfaces with buttons that perform certain procedures which are hidden from the user, creating a "black box" (Buchberger, 1990). Text-based programming is much less lenient and requires consideration of both the structure and the commands used to build a program. It is more difficult to build a program that solves a quadratic equation than use a solve command in another digital tool. The combination of understanding the different commands, what they do and how they interact together with understanding the structure of a program is demanding (Munthe, 2022a). The

second hindrance is the translation of mathematics into a programming code. The mathematical knowledge students possess of how to find the zero-point(s) of a polynomial function is often methodical and different from applying numerical methods. This translation from mathematics to the building of a program requires careful design, where the transition between the two are clear (Munthe, 2022a). The two lessons are examples of a "good" and "less good" implementation of programming into a mathematical classroom. A "good" implementation is recognized as containing more mathematical exploratory talk compared to adversity. A successful MPP design uses pre-existing knowledge and builds on that to facilitate the students' building of the program. Unsuccessful implementation comprises more adversity and less mathematical exploratory talk. Generally adversities are related to the syntax and structure of programming, which creates complexity and can hinder mathematical learning (Kirschner et al., 2018; Reiser, 2004). Difficulties such as syntax errors can be alleviated through an active teacher. When the MPP requires mathematics that the students are less familiar with, the complexity entails both programming and mathematics and can potentially create a double hurdle for the students to overcome (Kirschner et al., 2018; Ko et al., 2004; Reiser, 2004).

Concluding thoughts

From the discussion above, three main findings can be identified affecting the relevance of MPPs (1), the structure of the design (2, 3) and the choice of the mathematical area represented (2, 3).

1 When implementing programming into mathematics classrooms, it can facilitate mathematical exploratory talk.

2 Programming is best implemented to facilitate the in-depth learning of already-known mathematical concepts as this initiate exploratory talk, and care is needed when utilizing programming to learn new mathematical concepts.

3 Adversity is important and challenging when implementing programming into mathematics classrooms. It is important in that it can facilitate mathematical adidactical situations but challenging in that programming adds another layer of complexity.

The data show that programming can facilitate mathematical exploratory talk, but further research is needed to investigate whether programming can facilitate mathematical learning. This work investigated a small

number of student groups from one school, and research with a larger number of student groups from different schools is needed. Additionally, the role of the teacher was in not investigated.

References

Alrø, H. & Skovsmose, O. (2004). Dialogic learning in collaborative investigation. *Nordic Studies in Mathematics Education*, 9 (2), 39–62. https://ncm.gu.se/wp-content/uploads/2020/06/9_2_039062_alro.pdf

Benton, L., Hoyles, C., Kalas, I. & Noss, R. (2017). Bridging primary programming and mathematics: some findings of design research in England. *Digital Experiences in Mathematics Education*, 3 (2), 115–138. doi: 10.1007/s40751-017-0028-x

Berry, J. & Sahlberg, P. (2006). Accountability affects the use of small group learning in school mathematics. *Nordic Studies in Mathematics Education*, 11 (1), 5–31. https://ncm.gu.se/wp-content/uploads/2020/06/11_1_005032_berry.pdf

Bocconi, S., Chioccariello, A., Kampylis, P., Dagiené, V., Wastiau, P. et al. (2022). *Reviewing computational thinking in compulsory education: state of play and practices from computing education*. European Union. https://epublications.vu.lt/object/elaba:124209087/

Brousseau, G. (1997). *Theory of didactical situations in mathematics: didactique des mathématiques, 1970–1990*. Kluwer Academic.

Buchberger, B. (1990). Should students learn integration rules? *ACM Sigsam Bulletin*, 24 (1), 10–17. doi: 10.1145/382276.1095228

Bybee, R. W., Taylor, J. A., Gardner, A., Van Scotter, P., Powell, J. C. et al. (2006). *The BSCS 5E instructional model: origins and effectiveness*. BSCS. https://fremonths.org/ourpages/auto/2008/5/11/1210522036057/bscs5efullreport2006.pdf

Bråting, K. & Kilhamn, C. (2022). The integration of programming in Swedish school mathematics: investigating elementary mathematics textbooks. *Scandinavian Journal of Educational Research*, 66 (4), 594–609.

Calao, L. A., Moreno-León, J., Correa, H. E. & Robles, G. (2015). Developing mathematical thinking with scratch. In G. Conole, T. Klobučar, C., Rensing, J. Konert & E. Lavoué (Eds.), *Design for teaching and learning in a networked world* (pp. 17–27). Springer. doi: 10.1007/978-3-319-24258-3_2

Choi, J. & Walters, A. (2018). Exploring the impact of small-group synchronous discourse sessions in online math learning. *Online Learning*, 22 (4), 47–64. https://files.eric.ed.gov/fulltext/EJ1202373.pdf

Cobb, P., Boufi, A., McClain, K. & Whitenack, J. (1997). Reflective discourse and collective reflection. *Journal for Research in Mathematics Education*, 258–277. doi: 10.5951/jresematheduc.28.3.0258

Davidson, N. & Kroll, D. L. (1991). An overview of research on cooperative learning related to mathematics. *Journal for Research in Mathematics Education*, 22 (5), 362–365. doi: 10.5951/jresematheduc.22.5.0362

DePree, J. (1998). Small-group instruction: impact on basic algebra students. *Journal of Developmental Education*, 22(1), 2.

Djurdjevic-Pahl, A., Pahl, C., Fronza, I. & El Ioini, N. (2017). A pathway into computational thinking in primary schools. In T.-T. Wu, R. Gennari, Y.-M. Huang, H. Xie & Y. Cao (Eds.), *Emerging Technologies for Education. SETE 2016*. doi: 10.1007/978-3-319-52836-6_19

Drageset, O. G. (2014). Redirecting, progressing, and focusing actions – a framework for describing how teachers use students' comments to work with mathematics. *Educational Studies in Mathematics*, 85 (2), 281–304. doi: 10.1007/s10649-013-9515-1

Drijvers, P. (2015). Digital technology in mathematics education: why it works (or doesn't). In S. Cho (Ed.), Selected regular lectures from the 12th international congress on mathematical education. doi: 10.1007/978-3-319-17187-6_8

Francisco, J. M. & Maher, C. A. (2005). Conditions for promoting reasoning in problem solving: insights from a longitudinal study. *The Journal of Mathematical Behavior*, 24 (3-4), 361–372. doi: 10.1016/j.jmathb.2005.09.001

Fyfe, E. R. & Brown, S. A. (2020). This is easy, you can do it! Feedback during mathematics problem solving is more beneficial when students expect to succeed. *Instructional Science*, 48 (1), 23–44. doi: 10.1007/s11251-019-09501-5

Harel, G. & Sowder, L. (2005). Advanced mathematical-thinking at any age: its nature and its development. *Mathematical thinking and learning*, 7 (1), 27–50.

Hiebert, J. & Grouws, D. A. (2007). The effects of classroom mathematics teaching on students' learning. In F. Lester (Ed.), *Second handbook of research on mathematics teaching and learning* (pp. 371–404). Information Age.

Joubert, M. V. (2007). *Classroom mathematical learning with computers: the mediational effects of the computer, the teacher and the task* [PhD thesis]. University of Bristol. https://research-information.bris.ac.uk/en/studentTheses/classroom-mathematical-learning-with-computers-the-mediational-ef

Kazemi, E. & Stipek, D. (2009). Promoting conceptual thinking in four upper-elementary mathematics classrooms. *Journal of Education*, 189 (1-2), 123–137.

Kirschner, P., Sweller, J. & Clark, R. E. (2006). Why unguided learning does not work: an analysis of the failure of discovery learning, problem-based learning, experiential learning and inquiry-based learning. *Educational Psychologist*, 41 (2), 75–86.

Kirschner, P. A., Sweller, J., Kirschner, F. & Zambrano, J. (2018). From cognitive load theory to collaborative cognitive load theory. *International Journal of Computer-Supported Collaborative Learning*, 13 (2), 213–233. doi: 10.1007/s11412-018-9277-y

Knight, S. & Mercer, N. (2015). The role of exploratory talk in classroom search engine tasks. *Technology, Pedagogy and Education*, 24 (3), 303–319. doi: 10.1080/1475939X.2014.931884

Ko, A. J., Myers, B. A. & Aung, H. H. (2004). Six learning barriers in end-user programming systems. In *2004 IEEE Symposium on Visual Languages-Human Centric Computing*. doi: 10.1109/VLHCC.2004.47

Laborde, C. & Sträßer, R. (2010). Place and use of new technology in the teaching of mathematics: ICMI activities in the past 25 years. *ZDM*, 42 (1), 121–133. doi: 10.1007/s11858-009-0219-z

Lampert, M., Rittenhouse, P. & Crumbaugh, C. (1996). Agreeing to disagree: developing sociable mathematical discourse. In D. R. Olson & N. Torrance (Eds.), *The handbook of education and human development: new models of learning, teaching, and schooling* (pp. 731, 764). doi: 10.1111/b.9780631211860.1998.00032.x

Leung, A. & Baccaglini-Frank, A. (Eds.) (2016). *Digital technologies in designing mathematics education tasks: potential and pitfalls*. Springer. https://link.springer.com/content/pdf/10.1007/978-3-319-43423-0.pdf

Lou, Y., Abrami, P. C. & d'Apollonia, S. (2001). Small group and individual learning with technology: a meta-analysis. *Review of educational research*, 71 (3), 449–521.

Lv, L., Zhong, B. & Liu, X. (2023). A literature review on the empirical studies of the integration of mathematics and computational thinking. *Education and Information Technologies*, 28 (7), 8171–8193. doi: 10.1007/s10639-022-11518-2

Martin, L., Towers, J. & Pirie, S. (2006). Collective mathematical understanding as improvisation. *Mathematical thinking and learning*, 8 (2), 149–183. doi: 10.1207/s15327833mtl0802_3

Mercer, N. (2005). Sociocultural discourse analysis: analysing classroom talk as a social mode of thinking. *Journal of Applied Linguistics and Professional Practice*, 1 (2), 137–168. http://thinkingtogether.educ.cam.ac.uk/publications/journals/Mercer_JCL2005.pdf

Mercer, N., Dawes, L., Wegerif, R. & Sams, C. (2004). Reasoning as a scientist: ways of helping children to use language to learn science. *British educational research journal*, 30 (3), 359–377. doi: 10.1080/0141192041000168968 9

Mercer, N. & Littleton, K. (2007). *Dialogue and the development of children's thinking: a sociocultural approach*. Routledge.

Mercer, N. & Sams, C. (2006). Teaching children how to use language to solve maths problems. *Language and Education*, 20 (6), 507–528. doi: 10.2167/le678.0

Munthe, M. (2022a). Programming in the mathematics classroom – adversities students encounter. *Acta Didactica Norden*, 16 (4). doi: 10.5617/adno.9173

Munthe, M. (2022b). Press "run" to improve mathematical expertice (PRIME) [PhD thesis]. NMBU. https://hdl.handle.net/11250/3045236

Nathan, M. J. & Knuth, E. J. (2003). A study of whole classroom mathematical discourse and teacher change. *Cognition and instruction*, 21 (2), 175–207. doi: 10.1207/S1532690XCI2102_03

Noss, R. & Hoyles, C. (1996). *Windows on mathematical meanings: learning cultures and computers*. Springer Science & Business Media.

Park, I., Kim, D., Oh, J., Jang, Y. & Lim, K. (2015). Learning effects of pedagogical robots with programming in elementary school environments in Korea. *Indian Journal of Science and Technology*, 8 (26), 1–5.

Pedaste, M., Mäeots, M., Siiman, L. A., De Jong, T., Van Riesen, S. A. et al. (2015). Phases of inquiry-based learning: definitions and the inquiry cycle. *Educational Research Review*, 14, 47–61. doi: 10.1016/j.edurev.2015.02.003

Pirie, S. & Martin, L. (2000). The role of collecting in the growth of mathematical understanding. *Mathematics Education Research Journal*, 12 (2), 127–146. doi: 10.1007/BF03217080

Qian, Y. & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: a literature review. *ACM Transactions on Computing Education*, 18 (1), 1–24. doi: 10.1145/3077618

Reiser, B. J. (2004). Scaffolding complex learning: the mechanisms of structuring and problematizing student work. *The journal of the learning sciences*, 13 (3), 273–304. doi: 10.1207/s15327809jls1303_2

Resnick, L., Asterhan, C. S. & Clarke, S. (2017). Student discourse for learning. In G. E. Hall, D. M. Gollnick & L. F. Qzinn (Eds.), *Handbook of teaching and learning* (pp. X–Y). Wiley-Blackwell.

Resnick, L. B., Bill, V. & Lesgold, S. (1992). Developing thinking abilities in arithmetic class. In A. Demetriou, M. Shayer & A. Efklides (Eds.), *Neo-Piagetian theories of cognitive development: implications and applications for education* (pp. 210–230). Routledge.

Ryve, A. (2011). Discourse research in mathematics education: a critical evaluation of 108 journal articles. *Journal for Research in Mathematics Education*, 42 (2), 167–199. doi: 10.5951/jresematheduc.42.2.0167

Sfard, A. (2000). On reform movement and the limits of mathematical discourse. *Mathematical thinking and learning*, 2 (3), 157–189. doi: 10.1207/S15327833MTL0203_1

Shim, J., Kwon, D. & Lee, W. (2016). The effects of a robot game environment on computer programming education for elementary school students. *IEEE Transactions on Education*, 60 (2), 164–172. doi: 10.1109/TE.2016.2622227

Skolverket (2019). *Kursplanen i matematik*. https://www.skolverket.se/undervisning/gymnasieskolan/laroplan-program-och-amnen-i-gymnasieskolan

Slavin, R. E. (1990). Achievement effects of ability grouping in secondary schools: a best-evidence synthesis. *Review of educational research*, 60 (3), 471–499. doi: 10.3102/00346543060003471

Stein, M. K., Grover, B. W. & Henningsen, M. (1996). Building student capacity for mathematical thinking and reasoning: an analysis of mathematical tasks used in reform classrooms. *American Educational Research Journal*, 33 (2), 455–488. doi: 10.3102/00028312033002455

Tabach, M. & Schwarz, B. B. (2018). Professional development of mathematics teachers toward the facilitation of small-group collaboration. *Educational Studies in Mathematics*, 97 (3), 273–298. doi: 10.1007/s10649-017-9796-x

Toulmin, S. E. (1969). *The uses of argument*. Cambridge university press.

Urion, D. K. & Davidson, N. A. (1992). Student achievement in small-group instruction versus teacher-centered instruction in mathematics. *Problems, Resources, and Issues in Mathematics Undergraduate Studies*, 2 (3), 257–264. doi: 10.1080/10511979208965668

Utdanningsdirektoratet (2020). *Læreplan i matematikk 1.–10. trinn* (MAT1-05). https://www.udir.no/lk20/mat01-05

Yackel, E. & Cobb, P. (1996). Sociomathematical norms, argumentation, and autonomy in mathematics. *Journal for Research in Mathematics Education*, 27 (4), 458–477. doi: 10.5951/jresematheduc.27.4.0458

Notes

1 The bisectional method is a root-finding method applicable to any continuous function for which two values with opposite signs are known. The method consists of repeatedly bisecting the interval defined by the two values and then selecting the subinterval in which the function changes sign and that therefore contains the root (intermediate value theorem).

2 Also known as the Newton-Raphson method. It is a root-finding method with an initial guess (x_n) followed by the finding of the tangent line of the function at this point. The next estimate is where the tangent line crosses the x-axis. Algebraically, the estimate calculated can be expressed as

$$x_{n+1} = x_n - \frac{f(x_0)}{f\prime(x_0)}$$

Morten Munthe

Morten Munthe is a PhD fellow at Norwegian University of Life Sciences (NMBU). His research interests focus on digital technologies and discussions in mathematics education with emphasis on programming and how programming can contribute to mathematical discussion in the classroom.

morten.munthe@nmbu.no