Programming as a mediator of mathematical thinking: Examples from upper secondary students exploring the definite integral

Timo Tossavainen[1], Claes Johansson[2], Alf Juhlin[2] and Anna Wedestig[2]

[1] Luleå University of Technology, Sweden
[2] Luleå Upper Secondary School, Sweden

We report on three episodes from a case study where upper secondary students numerically explore the definite integral in a Python environment. Our research questions concern how code can mediate and support students' mathematical thinking and what kind of sociomathematical norms emerge as students work together to reach a mutual understanding of a correct solution. The main findings of our investigation are as follows. 1) Students can actively use code as a mediator of their mathematical thinking, and code can even serve as a bridge that helps students to develop their mathematical thinking collaboratively. Further, code can help students to perceive mathematical notions as objects with various properties and to communicate about these properties, even in other semiotic systems than the mathematical language. 2) For the participating students, a common norm was that an acceptable solution is a sufficient condition for the correctness of the solution method although students were aware of a problem in their code, yet also other norms emerged. This demonstrates that learning mathematics with programming can have an effect on what kind of sociomathematical norms emerge in classroom.

ARTICLE DETAILS

LUMAT General Issue
Vol 12 No 3 (2024), 78–99

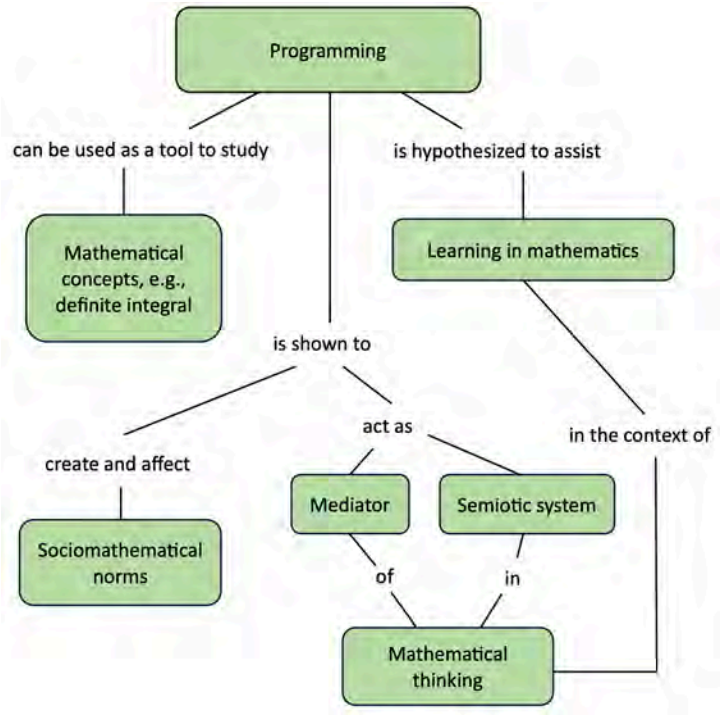
Received 19 February 2024
Accepted 6 May 2024
Published 15 May 2024

Pages: 22
References: 25

Correspondence:
timo.tossavainen@ltu.se

https://doi.org/10.31129/LUMAT.12.3.2155

Keywords: collaborative learning, definite integral, languaging, programming, sociomathematical norms



1 Introduction

In several European countries, including Sweden, programming has become a part of national school curricula in mathematics across all grade levels. The content areas in mathematics curriculum where programming is expected to be applied in Swedish schools include, e.g., algebra and problem solving. However, it has turned out that Swedish teachers face challenges in finding suitable ways to teach mathematics with programming. A survey of Kilhamn, Rolandsson, and Bråting (2021) indicates that the incorporation of programming into mathematics education has primarily resulted in teaching programming within a mathematical context. In other words, teaching with programming has not primarily focused on the learning of mathematics but merely on enhancing pupils' programming skills. One of the main reasons for that is that most mathematics teachers lack proper education in computer science and feel unsure and uncomfortable with programming (ibid., Johansson et al., 2023). Therefore, they are quite dependent on teaching materials that typically focus on the basic skills of programming rather than on learning mathematics with programming.

An examination of national curricula (e.g., Misfeldt et al., 2020) suggests that comparable findings could also emerge from other cultures. Indirect evidence of this phenomenon is present also in the literature review conducted by Forsström and Kaufmann (2018), where the positive impacts on student performance are frequently linked to general problem solving and logical reasoning rather than to any specific mathematical topics. Furthermore, geometry appears to be the content area where programming has been most frequently utilized (ibid.).

The purpose of the present case study is to examine the ways in which programming can support collaborative learning and students' communication in mathematics and students' mathematical thinking while they solve real mathematical tasks in small groups. Within a theoretical framework based on sociomathematical norms (Yackel & Cobb, 1996) and a multimodal languaging model (Joutsenlahti & Kulju, 2017), we investigate how programming makes mathematical thinking visual and how code can be used to share mathematical ideas. Our study is based on a teaching experiment conducted by the authors with a group of grade 11 students in the context of the definite integral.

Below we first introduce our theoretical framework and then review some previous studies relevant to ours. Following this, we state our research questions and present our method and results. Finally, we conclude this article with a discussion on our findings and conclusions.

2 Theoretical framework

Yackel and Cobb (1996) base their theory of sociomathematical norms on the recognized fact that mathematical learning involves both on active individual knowledge construction and acculturation into the mathematical practices of a society, such as a classroom of students and their teacher in a mathematics course. The aim of this theory is to furnish tools for analyzing students' activity in learning situations. The core concept of the theory is that, in distinction to general social norms that sustain classroom microcultures, there also are norms that are specific to the mathematical activities. These norms dictate, for instance, how students communicate with one another and with teachers about mathematical ideas and concepts. Likewise, what a group of students and their teacher deem an acceptable solution to typical a calculus exercise and how this differs from an acceptable solution to a typical exercise in geometry serves as another concrete example of these norms.

Originally, Yackel and Cobb concentrated on analyzing mathematical discussions – for example, explanation, justification, and argumentation – but sociomathematical norms also steer learners' other mathematical activities such as participation in small-group interactions (Yackel, Cobb, & Wood, 1991). It has also been noticed that sociomathematical norms and social norms are related to one another (Ozdemir Baki & Kilicoglu, 2023).

The development of sociomathematical norms obviously depends on a teacher's actions, e.g., on a teacher's noticing skills (ibid.), but they are not solely defined by a teacher but also students actively participate in defining them. They are always local by nature; two student groups with the same teacher can develop quite different kind of sociomathematical norms (Güven & Dede, 2017). Moreover, these norms are not constant but they evolve and change over time.

A learner's use of language in mathematics is inhetrently complex and multimodal, involving multiple modes, multiple presentations, and different types of written and oral texts (e.g., Morgan, Craig, Schuette, & Wagner, 2014; Moschkovich, 2021). The other theoretical perspective of this study, the multimodal languaging model of mathematical thinking, views the use of language and mathematical thinking as interplay between three semiotic systems or 'languages': the symbolic language of mathematics, natural language, and pictorial language (e.g., Joutsenlahti & Kulju, 2017). According to this model, students are able to express their mathematical thinking by utilizing the symbols of mathematics, their mother tongue or another natural language, and visual representations such as pictures, tangible devices, and diagrams.

A code is is neither an expression of any natural language nor typical use of mathematical symbols in school mathematics. However, it serves as an illustration of mathematical ideas. Therefore, it constitutes an element belonging to the third semiotic system in our theoretical model of languaging.

In the context of mathematics, the term "languaging" refers to the process of making meaning of and shaping knowledge about a mathematical idea through language, cf. Planas & Pimm (2024). In the process of multimodal languaging, a learner utilizes several semiotic systems, and communication between two or more individuals involves transmitting and receiving information through these channels. A common aim of research using this framework is to describe and analyze how and to what extent individuals utilize these semiotic systems in their communication, yet the ultimate goal of learning mathematics often entails that learners become proficient in communicating their mathematical thinking using the mathematical language and adhering to the sociomathematical norms of their community.

3 Previous research

Previous research indicates that upper secondary students often possess a superficial understanding about the definite integral concept (e.g., Rasslan & Tall, 2002; Attorps et al., 2010, 2013). While they may be able to recite the formal definition, many struggle to write meaningfully about it. For example, numerous students mistakenly associate it to the notion of area, leading to cognitive conflicts as the definite integral can also have negative values. Such prototypical images of a mathematical concept influence a learner's reasoning about the concept. Jones (2018) found that this is true specifically within the context of graphical representations of the definite integral.

In their literature review, Forsström and Kaufmann (2018) explored the utilization of programming in mathematics education. They categorized their findings according to three themes: the motivation to learn mathematics, students' performance in mathematics, and the collaboration between students and the changed role of the teacher. Somewhat surprisingly, they also remark (p. 28): "No deeper discussions occurred on the effect of collaboration, especially on students' mathematics learning. Furthermore, even if studies discuss collaboration as an important factor in students' learning, the learning is viewed in most studies as a change in individual knowledge instead of something that the group achieves as a group through collaboration". It proved challenging to find more recent studies that address this gap. Hence our study,

distinctly falling into the third category, aims to offer a fresh perspective by exploring how code can facilitate students' collaborative learning and communication.

A recent example of studies relevant to ours was conducted by Olsson and Granberg (2022) who investigated teacher–student interaction and its role in supporting 10–11 years old pupils' creative mathematical reasoning as they solved a geometry problem using Scratch. Their results suggest that, if a teacher poses appropriate questions targeting pupils' creative reasoning, it can help them to overcome some challenges related to learning mathematics through programming. Additionally, when pupils possess adequate programming skills, programming has the potential to support their reasoning and to enable the teacher to give timely feedback (ibid).

Olteanu (2022) explored the features of learners' reasoning and sensemaking as 13–14-year-old pupils familiarized themselves with the exterior angle concept in order to construct different regular polygons with Scratch. One of her findings is that, when designing appropriate tasks for teaching mathematics with programming, it is crucial to consider how connections are established between different concepts from mathematics and programming to promote sense-making and reasoning. In other words, her study contributes to understanding how to facilitate smoother transitions between the semiotic systems of the mathematical language and the programming language, although the theoretical framework of her study is not based on our perspective but rather on the notions of variation theory.

It appears that, particularly in the Nordic context, there are very few, if any, studies reporting on upper secondary students learning mathematics through programming. As mentioned in the introduction, this is consistent with the findings of Kilhamn et al. (2021); the use of programming has focused merely on solving problems in programming, not on solving mathematical problems with aid of programming. Therefore, the present study may provide some novel insights into this issue.

There are numerous studies examining the emergence of sociomathematical norms. Partanen and Kaasila (2015) offer a comprehensive summary of these studies, with their own investigation focusing on how these norms were negotiated during discussions among upper secondary students while investigating calculus in an inquiry-based collaborative teaching experiment. They found, for example, that the development of norms such as *'When investigating mathematics, one should approach the topic in a creative way'* is possible but it takes time and it challenging because students are used to solve mathematical tasks by following certain pre-given algorithms. Another example of the sociomathematical norms that Partanen and Kaasila (2015)

observed was *'Explicit justification in mathematics must be based on the properties of mathematical objects'*. What is particularly intriguing about this norm is its swift evolution during the teaching experiment conducted by one of the researchers, despite the participating students not being accustomed to justifying their mathematical claims in face-to-face interaction during lessons (ibid., p. 939). In other words, sociomathematical norms that can substantially enhance the quality of mathematical discourse, can evolve over a short period of time. There is no a priori reason to doubt that the introduction of programming in mathematics education could not have a comparable effect.

Recent studies on languaging have moved the focus “towards the communicative linguistic practices, with attention to social and interactional processes, rather than products of communication. This approach is contributing to revisiting mathematics teaching and learning as practices and processes in which language is a mediator and an agent of meaning” (Planas & Pimm, 2024, p. 135). A concrete example of the findings is that students develop their understanding about mathematical expressions and concepts in various ways. For example, Joutsenlahti and Kulju (2017) studied using the multimodal languaging model how primary students interpret symbolic expressions containing a division and a subtraction. Already the task '24/6 – 3' was languaged in three fundamentally different ways by a small group of students. In other words, although teaching can lead to that a group of students learn to write symbolic expressions syntactically correctly, these expressions are not necessarily read or understood in an unambigous way if students are not encouraged (e.g., Joutsenlahti & Tossavainen, 2018) to reveal and test their interpretations by languaging their mathematical thinking.

For a more thorough summary of recent studies on languaging and, more generally, language and communication in mathematics education, see, e.g., Morgan et al. (2014), Moschkovich (2021), and Planas and Pimm (2024).

4 Design and research questions

In this case study, we survey how code can mediate students' mathematical thinking and what kind of sociomathematical norms emerge when a group of upper secondary students numerically explores the definite integral using programming. Our study also possesses an ethnographic dimension as three of the authors are teachers of the participating students and know them well. This close relationship has greatly aided us in interpreting our data.

Regarding sociomathematical norms, we are particularly interested in those influencing students' reasoning as they evaluate the correctness of their solutions to integration problems. Additionally, we aim to investigate students' languaging of their mathematical thinking and the role of code in languaging as students explore the definite integral with a set of test functions. Our explicit research questions are as follows.

1. How does code mediate students' mathematical thinking and influence their languaging when studying the definite integral with programming?
2. What kind of sociomathematical norms emerge when students negotiate different solution methods and establish a shared understanding of a correct solution while studying the definite integral with programming?

5 Method

Data for the present study were collected at an upper secondary school in Northern Sweden during one lesson. The participating students are enrolled in a mathematically emphasized study program and they had already encountered the concept of the definite integral. They possessed experience with programming in Python to that extent that they were able to read and edit code without any significant problems. Figure 1 illustrates an example of the code used in this study.

```
9  from pylab import *
10 def f(x):
11     return sin(x)/x
12
13 def myIntegral(a,b,n):
14     h = (b-a)/n
15     I = 0.0
16     x = 0.0
17     for i in range(n):
18         I += f(x)*h
19         x += h
20     print(I)
21
22 myIntegral(0,1,100)
```

Figure 1. An example of the code used in the study.

As Figure 1 shows, students were numerically investigating the integral $\int_a^b f(x)dx$. The integrand is defined on line 11 and the bounds of integration along with the density of partition on line 22.

Twelve students worked in four small groups with each one consisting of 2–4 students. In addition to the provided code, they were given a list of test functions. Their task was to examine how the value of Riemann sum for different functions varies with the length of the sub-interval, thereby developing an understanding of the definite integral as a limit of Riemann sums.

The activities of the small groups were video-recorded, capturing the screen view of the laptop used by the group and the voices of the group members, while not including students' faces or physical actions. We conducted content analysis (e.g., Krippendorff, 2018) to analyze the discussions of each group. By repeatedly watching the videos, we identified several interesting episodes which were transcribed for the analyses. We deemed an episode interesting, e.g., when students encountered a mathematical problem or another question they could not immediately interpret or solve correctly. Predictably, our data also encompass episodes where students' attention is drifted away from the mathematical problem or they became more engrossed in programming than the mathematical tasks. These episodes are also noteworthy. However, for this article, we chose to concentrate on episodes demonstrating the code's potential to mediate mathematical thinking, and where we observed an influence of a sociomathematical norm that we deemed worthy of reporting. Following a thorough discussion, three episodes were selected.

The content analysis of the selected episodes involved systematic reading of the transcripts, with a focus on uncovering the key inferences made by students. The interpretation of the episodes was initially conducted by the first author and subsequently validated or adjusted through collaborative discussion.

There are some limitations to the present study. Firstly, the absence of video recordings capturing students' behaviour makes interpreting their activities somewhat challenging. Specifically, we were unable to gauge the level of attention each student devoted to the details of the code; instead, we could follow only the voices and the movements of the cursor. Additionally, the quality of laptops' microphones could have been better, as there were instances where background noise from other groups working in the same classroom made it difficult to discern what students were saying from the audio recordings. However, each episode was reviewed by multiple researchers

who also observed the lesson, providing confidence in the accuracy of the transcriptions.

Another limitation pertains to the generalizability of our results. Obviously, a small sample is rarely representative; therefore, our study can only bring up examples of the studied phenomena and not answer to the question how usual or regular these phenomena are. On the other hand, there is no need to generalize our findings as sociomathematical norms are *per se* local; in another classroom, an observer could find different kind of norms guiding students' activities. Consequently, our study like any other investigation into sociomathematical norms can only demonstrate the types of norms that exist but not say how typical they are in a larger population.

Regarding ethical considerations, all participants were over fifteen years old which in Sweden grants them the autonomy to decide whether to participate in the study. All students participated voluntarily, and no personal or sensitive information about them was asked or recorded. However, since three of the researchers are also the students' teachers, they recognize the students by their voices. The video recordings are stored according to the guidelines of the affiliations represented by the authors and will be not made available to third parts in order to preserve the participants' anonymity.

6 Results

A general observation from the lesson was that the use of code appeared to assist students in maintaining focus on the given task, despite a few occasional moments when they discussed unrelated topics. Students in small groups focused on discussing on what they saw on the screen of a laptop. Moreover, it was clear by students' discussion that they understood how the provided code was supposed to work and how it was related to the mathematical task.

Next we report on three episodes in detail. Since sociomathematical norms and the mediating effect of code can be observed only in situ, we address both research questions within each episode and summarize our findings at the end of the section.

6.1 Episode 1

Here four students are first studying $\int_0^1 f(x)dx$ for $f(x) = x$ with $n = 10$ and the program prints out the answer 0.4500.... In the beginning, one student codes and two students comment on what is happening on the screen. They notice immediately that

the answer is not exactly what they expected (0.5). One of them suggests that this depends on the roughness of the division of the interval and refers to the code by saying that increasing the value of n to one hundred should already give a better estimate for the correct solution. The coding student performs this change and runs the code. The given answer 0.49500... satisfies everyone in the group, so, they arrive at a shared view of the solution quicky and easily. Here the code plays a central role in mediating students' mathematical thinking as the change of the value of one parameter was a visible operation and everyone agreed on the correctness of the reasoning behind this operation.

Next, students begin to examine $f(x) = e^{x^2}$ with $n = 10$, arriving at a negative value -1.381... One of the students observes then that something is wrong as he realizes that f is a positive function and thereby the value of the Riemann sum should be positive, too. Quite naturally, they try to solve the observed mathematical problem by performing the same operation as previously: replacing $n = 10$ first by $n = 100$ and then by $n = 10\,000$. However, they still arrive at a negative answer.

Now, an interesting turn follows. One of the students suggests that they should test their code with a very simple function $f(x) = 7$. They do so.

> It should be exactly seven, independently how large [the value of n is]. (Student A)

They run the code.

> Yes. (Student A)
>
> Or... but why it becomes minus seven? (Student B)
>
> I think that it becomes minus...in the error... wait! (Student A)

The cursor points to the parameter n and there is silence for about twelve seconds. Then the cursor is moved onto line 14 where parameter h is defined.

> Yes, it should be b minus a. (Student A)

Both discussing students observe that they had inserted this line incorrectly: $h = (a - b)/n$ instead of $h = (b - a)/n$. Then a student who has been silent so far says:

> Has it gone wrong also in the other [previous cases with other functions]? (Student C)

Student A agrees on that the code was erroneous also when they ran it for the previous functions. Now they run the corrected code both for $f(x) = x$ and $f(x) = e^{x^2}$ and get the positive results. Seeing this convince everyone that they have arrived at a correct solution.

Interestingly, students do not discuss in any way, why the incorrect code gave correctly a positive result in the case of $f(x) = x$ although they now are aware of its incorrectness, cf. the comment above made by Student C. Indeed, this episode ends as students notice that the corrected code gives positive values for the Riemann sums in the case of $f(x) = e^{x^2}$.

The sociomathematical norm that steers students to accept their solution appears to be satisfied when the code gives a value which they can consider an acceptable approximation of the value of the definite integral they are studying. In this particular case, already the positivity of the value can play a decisive role.

When it comes to languaging, already the beginning of the episode reveals that students can move from one semiotic system to another and back quite smoothly. Their communication happens mostly within the semiotic systems of pictorial language (code) and natural language (Swedish). Students hardly mention any other mathematical term during the episode than values of the definite integrals and their signs, yet they can effectively communicate the essential mathematical ideas needed for discussing the task, the correctness of solution and, especially, for correcting the error in their code. This does not mean that students did not use also the semiotic system of mathematical language. On the contrary, they must have used that when they interpreted the task in the beginning of the episode. Further, when students noticed the error in their code, it was possible only because they could read and comprehend that the values of $f(x) = e^{x^2}$ are positive. So, a semiotic system can play an important role in the languaging process although its signs remained unspoken.

A reason for smooth transitions between the semiotic systems appears to be that all of them can read both mathematical language and the code fluently and they do not have to interrupt their communication in natural language, for example, to explain one another on which line the division of the integration interval is adjusted.

On the other hand, although students appear to master all three semiotic systems well enough to discuss the integration problem, they fail to observe that an erroneous code gave them a correct answer in the case of $f(x) = x$. This demonstrates two things. First, even advanced languaging skills does not always guarantee that mathematical thinking were correct. Second, sociomathematical norms guide students'

mathematical thinking quite powerfully. In this group, a consequence of the norm '*a seemingly correct answer guarantees the correctness of the solution method*' was that students did not experience any need to revise their solution in spite of the fact that they became aware of an error in their code.

6.2 Episode 2

In this episode, two students are studying the definite integral of $f(x) = sin(x)\ /x$ and the programming environment gives the following message as the students run their code: "*RuntimeWarning: invalid value encountered in double scalars. return sin(x)/x*". Obviously, this is due to the fact that the interval is $[0, 1]$ and the given function is not defined at $x = 0$. However, students do not observe this from the beginning.

> Wait... What? ... Invalid value? ... I get a message on an error. Although it is not an error, it prints out an incorrect thing [meaning: something else than expected] (Student D)
>
> What? (Student E)
>
> Yes, I get some 'RuntimeWarning'. What [does it mean]? (Student D)

They run the code a couple of times and get the same message over and over again.

> Ask X [teacher's name]. (Student E)
>
> X, if one gets RuntimeWarning on the second [window where the code is run] (Student D)

Teacher X comes and advices students to focus on the function f. Student D speaks about the present function, but Student E starts speaking about the previous case $f(x) = 7$ and what happened then for a large value of n.

> So, I put too many nines in the previous, for seven, so it will take a long time to load... (Student E)

Student D says something indicating that it was so in the previous case and continues then

> How does [the curve of] $sin(x)$ over x look like? (Student D)
>
> It goes to... some value... It goes to zero when it grows. (Student E)

Both students start to think about this and it takes about 45 seconds.

> OK, it becomes like... becomes like... It becomes smaller and smaller. (Student D)
>
> Yes... It does not divide at zero. (Student E)
>
> No. So, at zero it is not defined. Is it so that it is not defined at zero and hence one must choose, say, zero point... zero point zero zero one. Does it make any difference? (Student D)

Now Student D runs the code by calling $minIntegral(0.00001, 1{,}10)$ and gets again the same runtime warning. For a couple of seconds she is surprised but notices then that she needs to adjust parameter x on line 16.

> No, what if one changes [the value of x] over here. (Student D)
>
> Then it will work. (Student E)
>
> Yes. Hm, for that x can't be... Yes, of course, it is clear. It is still from zero to one, but x can't be zero... It is almost from zero to one... Then the sum is 0.593. (Student D)

In this episode, Student D is seemingly more knowledgeable than Student E. When they meet a problem with the runtime warning and ask help from their teacher, Student E gets lost and starts to talk about something that is not relevant for solving the problem with the runtime warnings. Student D listens to him but draws his attention back to the issue by referring to the present function in the code. Student E is still a bit lost as he considers how $sin(x)\,/x$ decreases when x grows, but then – without neither of them saying something – they start to see that the essential question is what happens when x goes to zero. It appears that seeing the code is just what helps both of them to observe this.

It is crucial to note that, while students engage in a discussion about the properties of the integrand $f(x) = sin(x)\,/x$, they remain aware of their principal objective: to compute $\int_0^1 f(x)dx$. We motivate this claim by the fact that students recognized the undefiniability of $f(x)$ only after having run the code multiple times. Thus, the code now plays a pivotal role in mediating students' mathematical thinking, even though they do not explicitly reference code in their discourse. However, both the code and the runtime warning are constantly visible to them, which makes it easier (compared to a situation where they would not see the code) for them to focus on searching for

the reason for the error by exploring different alternatives: a flaw in the integrand, interval, etc. And they succeed in finding the error's source and then go back to computing an approximation of $\int_0^1 f(x)dx$.

Another interesting observation is that there is a clear qualitative difference in students D's and E's languaging. Student D uses the mathematical language more actively and coherently, whereas Student E builds more on the other two semantic systems, e.g., by using the natural language word 'it', referring to things which are not exactly correct in that context. However, with aid of the code, he can follow the reasoning of D and vice versa, especially, when they solve the issue related to the starting point of the interval.

Also in this episode, we notice a similar sociomathematical norm as in the previous episode: students appear to be happy with their solution if the value of the Riemann sum given by the code is plausibly what they expected it to be. They do not start a discussion about the preciseness of the approximation but they focus more on confirming that the code really corresponds to the algorithm of computing Riemann sums in the case where the given function is not defined in the starting point of the interval.

6.3 Episode 3

In this episode, two students are studying $\int_0^1 f(x)dx$ with $f(x) = e^{x^2}$. This episode is especially interesting because students' discussion happens mostly using very informal natural language and the code. However, both of them seem to be able to follow each other's mathematical thinking carefully and correctly. Another difference compared to the previous episodes are that students work very tightly together and, in some places, where one of them starts a sentence, the other student may respond immediately and continue the same sentence meaningfully, e.g.,

> Now we are going to do something cool... (Student F)
>
> ... the cool thing. (Student G)
>
> e−x [F moves the cursor to line 11] (Student F)

Student G reacts immediately before F has written something:

> One can write it in two different ways. (Student G)
>
> Write it in the cool way. You can. (Student F)

> The real cool way? (Student G)
>
> The extreme cool way. (Student F)

Student G writes first “e^x^2” on the correct line and the discussion continues immediately:

> Can one use that...? (Student F)
>
> Don't know. Let's see... I don't believe that it works... (Student G)
>
> ...then with stars. (Student F)

Discussion continues intensively in this style and students observe quickly that they also need parenthesis. They test two different ways to write parenthesis running the code with $n = 100$. The result is 99,0266...

> Does it give the same value? Bugger! Oh my god, it is the same value. But we should now have ten as that one is big [meaning the value of n] (Student F)
>
> Shall I reduce to ten? [replaces $n = 100$ by $n = 10$] (Student G)

Students run the code with $n = 10$ resulting now to the value 9,2247...

> It disturbs me that that one goes down. Now we shall change the function to seven [meaning $f(x) = 7$] (Student F)

Student G does not reply to the suggestion made by F but starts to analyze the code carefully. Student F accepts this and they together focus on the code as G moves the cursor through the code.

> Are you sure that you wrote the whole.. [meaning did F copy the original code correctly from the paper where the task was given] (Student G)
>
> Yes, look here...[a short pause] I agree, there is something strange here (Student F)

First students search for an error on line 14, but they notice that it is correct. Then F focuses on line 17, where they should have '*for i in range(n)*' but they have '*for I in range(n)*'.

> Wait, what happens if one changes that one? [F changes i to I and runs the code] (Student F)

Why do you have the big i and the small i? (Student G)

Because they have [meaning the code in the formulation of the task], that one is the big i and that one is the small i, aren't they? (Student F)

Hm..[being somewhat unsure] (Student G)

What? Look, here is the big i...(Student F)

Yeah [now agreeing] (Student G)

... and here is the small i. (Student F)

The episode continues so that students run their revised code with $f(x) = x$ and become confident that the code is now correct as they receive a plausible result. Thereafter, they return to studying $\int_0^1 f(x)dx$ with $f(x) = e^{x^2}$.

Also this episode shows that students' mathematical communication can be effective although they actively avoid using the mathematical language in their speech. It takes only 2 minutes and 30 seconds from the beginning of the episode to the point where they test the revised code with $f(x) = x$. In our view, a contributing factor for this effectivity was that they understood one another very well. Already the use of informal expressions helped them to focus on the same details while solving the task.

On the other hand, both of them focused carefully on the code. This became visible especially in those occasions where one of them started saying something or editing the code and the other student reacted to this immediately. So, code does not only mediate the mathematical thinking of one student to another student but it can also bridge their mathematical thinking together. As the previous episode, the code appears to have played a fundamental role in enhancing their mathematical communication. Students were actively engaged in the integration task and the visible code enabled them to concentrate on details of the task. They responded quickly and intensively, e.g., to the anticipated properties of the outcome. Such communication would be considerably challenging without code mediating their mathematical ideas and intentions. A prerequisite for this is that students appeared to have a good understanding about the task which was given in the mathematical language. For example, although they did not use mathematical terms in their speech, they noticed quickly both for $(x) = e^{x^2}$ and $f(x) = x$ whether the value of the definite integral was plausible or not. Also this contributed to that students could communicate their observations meaningfully using the other two semiotic systems than the mathematical language.

Not surprisingly, also these students adhered to the already-mentioned socio-mathematical norm: if the answer is plausible, also the solution method is accepted being correct.

6.4 Summary

We present a synthesis of our findings from the three episodes discussed above, along with our answer to the research questions as follows.

1) Across all episodes, it became evident that students effectively utilized code as a mediator of their mathematical thinking when explaining their mathematical ideas to one another. In Episode 3, we observed that code could serve as a bridge to facilitate collaborative development of mathematical thinking. Further, we noted that code helped students to perceive mathematical notions as objects or schemes with various properties, cf. Arnon et al. (2014). For instance, students were able to conceptualize the algorithm for computing a Riemann sum as an independent entity and discuss its properties. An example of this is that students anticipated a positive value for the definite integral *already before* running the code. In each episode, students engaged with all three semiotic systems of languaging, but there were notable differences among individuals and groups in the extent to which they relied on each semiotic system. In Episode 3, students exhibited effective and advanced mathematical thinking despite primarily communicating without using mathematical language.

2) A common sociomathematical norm noticed in each episode was that an acceptable solution also justified the solution method, even though in some situations students were aware of problems related to their code. Another norm was that when students argued for their ideas they relied on the properties of mathematical objects, e.g., the positivity of a function or the decreasing value of the definite integral, cf. Partanen and Kaasila (2015). Furthermore, we observed that the use of programming could also have an effect on the development of sociomathematical norms. For example, students demonstrated persistence when they noticed that their code did not work correctly for a given function. Rather than giving up, they tested the code with another function. This happened both in Episode 1 and Episode 3. Our experience suggests that such behaviour is less common in a traditional learning situation with analogous problems.

7 Discussion

In our experiment, students demonstrated the ability to smooth transitions between the semiotic systems of code, natural language, and the symbolic language used in the formulation of the task. Our particular task was to explore how code, a component of pictorial language, can mediate mathematical thinking. Our findings, especially, those related to Episodes 2 and 3, suggest that code as a component of pictorial language can play a dual role. On the one hand, code as a singular element kept students mindful of their task, sparing them from repetitive discourse on the need to compute definite integrals. On the other hand, code as a detailed breakdown of steps necessary to solve the computational task not only aided students in detecting issues, such as when the integrand is not defined across the entire interval, but also enhanced their overall understanding.

Moreover, it was delightful to observe that students were able to discuss the properties of the concept of the definite integral (e.g., the positivity of result) within the semiotic system of code. This indicates a high level of understanding of the concept, cf. Arnon et al. (2014). We interpret this achievement primarily as a result of students' proficiency in reading and understanding their code. Two indications of this are the facts that students were able to correct the small errors in their codes quite quickly and their adept discussion of code details. These findings align well with those made by Olsson and Granberg (2022) who remarked that sufficient programming skills are a prerequisite for programming to support pupils' mathematical reasoning. Using the terminology of the present paper, both their findings and ours suggest that code can be a mediator of mathematics thinking if the participants in a mathematical discussion possess a strong command of the programming language's semiotic system.

In all episodes, we observed also some other sociomathematical norms that reported above; for example, the norm also noted by Partanen and Kaasila (2015): students negotiate the correctness of their solution by referring to some properties of the mathematical object they are studying. In the first episode, it was the positivity of the value of Riemann sum, in the second and third episodes, it was the expectation that the size of the value of Riemann sum would match their anticipation.

Another norm observed by us, which we have not found in the previous studies from the Nordic context, is that students were satisfied with their solution method if they were satisfied with the solution given by the method even when they knew that they had an error in their code. A case study cannot definitely explain how this norm developed but a plausible explanation is that this kind of reasoning is valid in most

cases; for instance, the probability of two random errors in computations offsetting each other is low.

Based on our extensive experience with teaching mathematics, if a student is not sure about her solution method while solving a problem by hand, she quite rarely tests her method for another similar problem. Indeed, in a traditional mathematics lesson, students often stop working and wait for a teacher or another student to solve the problem if they encounter an obstacle themselves. For instance, consider how students would have developed a shared understanding of their solution if the task had been to compute the Riemann sums by hand. Would the group in the first episode have tested their algorithm for the function $f(x) = 7$ when they observed that it gives an impossible value for the function $f(x) = e^{x^2}$? We believe that in such a situation, students most often focus on finding an error in the computations related to the given task, and they only very rarely move on to testing their method on a different task. Now, in the first and third episodes, students encountered an error in their code, and, after discovering it, they went on correcting the code by first testing it with a very simple function and then re-running the revised code with another test functions. We interpret this as an evidence for that programming can steer what kind of norms emerge or alter existing norms, prompting students to become more active to make hypotheses and testing them, rather than adopting a passive role in a learning situation.

An interesting question not addressed in this study is whether teaching mathematics with programming significantly increases students' motivation to study mathematics. It is not self-evident that the use of digital technology in mathematics education leads to increased motivation (e.g., Drijvers, 2018; Tossavainen & Faarinen, 2019). However, the participating students were engaged in solving the given task with aid of code and they actively discussed their ideas. Therefore, we can conclude that, at the very least, the use of programming was not an hindrance to students' engagement in our experiment.

Furthermore, one of us conducted the same lecture as reported in this paper with another group of students who had more experience with programming. An interesting observation from this session was that students' discussions were more mathematically focused. Again, it appears that the level of students' programming skills has a significant effect on the extent to which the use of programming can enhance their learning in mathematics, as noted by Olsson and Granberg (2022).

Finally, in the third episode, the tendency to avoid the mathematical language while solving the task did not indicate any problems in mathematical thinking. On the contrary, the use of pictorial language only supported students' discussion on a strategy for solving the problem with non-plausible values of the definite integral. In our opinion, this indicates that a natural language or a pictorial language are well-suited for communicating problem solving strategies and applying other metacognitive skills.

8 Conclusions

In conclusion, the above discussion on the sociomathematical norms observed in this study suggests that that programming can have a positive effect on the emergence and development of sociomathematical norms in classroom. Additionally, it can help students to perceive mathematics as a more dynamic and evolving discipline than it is often seen, cf. Tossavainen et al. (2020). By a positive effect, we mean that students can become more active in formulating their own hypotheses and testing them, which is a fundamental problem-solving skill and an attitude associated with a dynamic mathematical mindset.

However, it is important to note that the use of programming does not automatically ensure that students pay a sufficient attention to their solution methods, even though code makes the method both visible and testable. We conclude that the norm that a correct solution justifies the solution method is deeply rooted in the realm of learning mathematics.

On the other hand, while our findings demonstrate that the use of programming in teaching and learning mathematics can enhance collaborative working methods and facilitate the mediation of mathematical ideas, we conclude that programming does not necessarily make learning or teaching easier. Instead, it imposes high demands on all participants' abilities to use both mathematical and programming languages. In particular, teachers may find their knowledge of programming languages insufficient, which can affects how fluently they can transition between mathematical and programming languages, cf. Olteanu (2022), Kilhamn et al. (2021), and Johansson et al. (2023).

In line with findings of Olteanu (2022) and Olsson and Granberg (2022), we also conclude that a task that works well in a traditional teaching of mathematics may not necessarily translate well into a programming environment. A task may need to be rephrased to help learners to better understand the relationship between inherent

mathematical notions and their counterparts in a programming language. Therefore, it is essential to pay attention to developing new kinds of learning materials for teaching mathematics with programming and not only invest on computers and access to programming environments if we want that this shift in teaching paradigms to be successful, cf. Kilhamn et al. (2021) and Johansson et al. (2023).

References

Arnon, I., Cottrill, J., Dubinsky, E., Oktaç, A., Roa Fuentes, S., Trigueros, M., & Weller, K. (2014). *APOS Theory. A framework for Research and Curriculum Development in Mathematics Education*. Springer. https://doi.org/10.1007/978-1-4614-7966-6

Attorps, I., Björk, K., Radic, M., & Tossavainen, T. (2010). The learning study model and the teaching of the definite integral concept. In M. Asikainen, P. E. Hirvonen, & K. Sormunen (Eds.), *Ajankohtaista matemaattisten aineiden opetuksen ja oppimisen tutkimuksessa. Matematiikan ja luonnontieteiden opetuksen tutkimuspäivät Joensuussa 22.-23.10.2009. Reports and Studies in Education, Humanities, and Theology* (pp. 77–86). University of Eastern Finland.

Attorps, I., Björk, K., Radic, M., & Tossavainen, T. (2013). Varied ways to teach the definite integral concept. *International Electronic Journal of Mathematics Education, 8*(2–3), 81–99.

Drijvers, P. (2018). Empirical evidence for benefit? Reviewing quantitative research on the use of digital tools in mathematics education. In L. Ball, P. Drijvers, S. Ladel, H-S. Siller, M. Tabach, & C. Vale (Eds.), *Uses of technology in primary and secondary mathematics education* (pp. 161–175). Springer. https://doi.org/10.1007/978-3-319-76575-4_9

Forsström, S. E., & Kaufmann, O. T. (2018). A literature Review exploring the use of programming in mathematics education. *International Journal of Learning, Teaching and Educational Research*, *17*(12), 18–32. https://doi.org/10.26803/ijlter.17.12.2

Güven, N. D., & Dede, Y. (2017). Examining social and sociomathematical norms in different classroom microcultures: Mathematics teacher education perspective. *Educational Sciences: Theory & Practice*, *17*(1), 265–292. https://doi.org/10.12738/estp.2017.1.0383

Johansson, C., Juhlin, A., Tossavainen, T., & Wedestig, A. (2023). Nyfikenhet och tillräcklighet. Gymnasielärares erfarenheter av att undervisa matematik med programmering. *Nämnaren, 49*(3), 35–41.

Jones, S. R. (2018). Prototype images in mathematics education: the case of the graphical representation of the definite integral. *Educational Studies in Mathematics, 97*(3), 215–234. https://doi.org/10.1007/s10649-017-9794-z

Joutsenlahti, J., & Kulju, P. (2017). Multimodal languaging as a pedagogical model—A case study of the concept of division in school mathematics. *Education Sciences, 7(*1), 9. https://doi.org/10.3390/educsci7010009

Joutsenlahti, J. & Tossavainen, T. (2018). Matemaattisen ajattelun kielentäminen ja siihen ohjaaminen koulussa. In J. Joutsenlahti, H. Silfverberg & P. Räsänen (Eds.), *Matematiikan opetus ja oppiminen* (pp. 410–430). Niilo Mäki Instituutti.

Kilhamn, C., Rolandsson, L., & Bråting, K. (2021). Programmering i svensk skolmatematiken? *LUMAT – International Journal on Math, Science and Technology Education, 9*(1), 283–312. https://doi.org/10.31129/LUMAT.9.2.1457

Krippendorff, K. (2018). *Content analysis: An introduction to its methodology* (Fourth edition). Sage. https://doi.org/10.4135/9781071878781

Misfeldt, M., Jankvist, U.T., Geraniou, E., & Bråting, K. (2020). Relations between mathematics and programming in school: Juxtaposing three different cases. In A. Donevska-Todorova, E. Faggiano, J. Trgalova, Z. Lavicza, R. Weinhandl, A. Clark-Wilson & H-G. Weigand (Eds.), *Proceedings of the 10th ERME topic conference on mathematics education in the digital era (MEDA 2020)*, (s. 255–262). Johannes Kepler University.

Morgan, C., Craig, T., Schuette, M., & Wagner, D. (2014). Language and communication in mathematics education: An overview of research in the field. *ZDM*, *46*(6), 843–853. https://doi.org/10.1007/s11858-014-0624-9

Moschkovich, J. N. (2021). Learners' language in mathematics classrooms: What we know and what we need to know. In N. Planas, C. Morgan & M. Schütte (Eds.), *Classroom research on mathematics and language: Seeing learners and teachers differently; classroom research on mathematics and language: Seeing learners and teachers differently* (pp. 60–76). Routledge/Taylor & Francis Group. https://doi.org/10.4324/9780429260889-5

Olsson, J., & Granberg, C. (2022). Teacher-student interaction supporting students' creative mathematical reasoning during problem solving using Scratch. *Mathematical Thinking and Learning*, Advance online publication. https://doi.org/10.1080/10986065.2022.2105567

Olteanu, C. (2022). Programming, mathematical reasoning and sense-making. *International Journal of Mathematical Education in Science and Technology*, *53*(8), 2046–2064. https://doi.org/10. 1080/0020739x.2020.1858199

Ozdemir Baki, G., & Kilicoglu, E. (2023). Social and socio-mathematical norms constructed by teachers in classes through the development of noticing skills. *International Electronic Journal of Mathematics Education*, *18*(1), em0723. https://doi.org/10.29333/iejme/12649

Partanen, A. M., & Kaasila, R. (2015). Sociomathematical norms negotiated in the discussions of two small groups investigating calculus. *International Journal of Science and Mathematics Education*, *13*(4), 927–946. https://doi.org/10.1007/s10763-014-9521-5

Planas, N. & Pimm, D. (2024). Mathematics education research on language and on communication including some distinctions: Where are we now? *ZDM – Mathematics Education*, *56*, 127–139. https://doi.org/10.1007/s11858-023-01497-0

Rasslan, S., & Tall, D. (2002). Definitions and images for the definite integral concept. In A. Cockburn & E. Nardi (Eds.), *Proceedings of the 26th International Conference for the Psychology of Mathematics Education* (Vol. 4, pp. 89–96). Norwich, UK.

Tossavainen, T., & Faarinen, E. C. (2019). Swedish fifth and sixth graders' motivational values and the use of ICT in mathematics education. *Eurasia Journal of Mathematics, Science and Technology Education*, *15*(12), em1776. https://doi.org/10.29333/ejmeste/108533

Tossavainen, T., Rensaa, R. J., Haukkanen, P., Mattila, M., & Johansson, M. (2020). First-year engineering students' mathematics task performance and its relation to their motivational values and views about mathematics. *European Journal of Engineering Education*, *46*(4), 604–617. https://doi.org/10.1080/03043797.2020.1849032

Yackel, E., & Cobb, P. (1996). Sociomathematical norms, argumentation, and autonomy in mathematics. *Journal for Research in Mathematics Education*, *27*, 458–477. https://doi.org/10.2307/749877

Yackel, E., Cobb, P., & Wood, T. (1991). Small-group interactions as a source of learning opportunities in second-grade mathematics. *Journal for Research in Mathematics Education*, *22*(5), 390–408.