Programming to learn mathematics – exploring student teachers' instrumental genesis

Johanna Pejlare and Laura Fainsilber
Chalmers university of Technology and University of Gothenburg

Programming is now a prescribed part of the curriculum in mathematics in both primary and secondary school in Sweden, as in many other countries. Teacher training must thus prepare students for the challenges of teaching mathematics with programming. We explore how student teachers see their own training in programming in relation to mathematics and what opportunities to learn mathematics they believe that programming can offer pupils in school. An instrumental approach is used to analyse observations and a questionnaire on secondary school student teacher' experience of a programming lab, where they investigate Riemann sums with programming. We find that students feel challenged by both the programming and the mathematical content, and that they see the challenges as useful, both for themselves and for their future pupils.

Introduction

Technological and digital competence have become extremely important in society and have an increasing impact on people's lives. As a consequence, programming has been introduced in the school curriculum in many countries. In Sweden, programming content was added to the national mathematics curriculum for compulsory school and upper secondary school in 2018 and reformulated in 2021. In the syllabus for the advanced mathematics courses at upper secondary school, programming is included as *a tool in problem solving, data processing or application of numerical methods* (Skolverket, 2011). An important factor for the successful integration of programming into mathematics education is the teachers' subject knowledge. However, today, there is a gap between the need for programming in school and what many teachers feel their knowledge is in programming (Misfeldt, Szabo & Helenius, 2019). To bridge this gap and meet the new curricula, a programming strand is included in the secondary school mathematics teacher program at the University of Gothenburg. There is no course in programming per se. Instead, programming assignments are integrated in all the mathematics courses.

The present study is a part of a larger project aimed at exploring the potential for learning mathematics through programming (Pejlare, 2021). In particular, we study how programming can be integrated in mathematics education, both at upper secondary school and in the teacher education program. In this study we focus on students preparing to teach mathematics at the secondary school level. The aim is to investigate the student teachers' understanding of their own learning of mathematics through programming, as well as their understanding of the prospects for programming as a tool

in mathematics education in secondary school. To meet this aim, we use observations of secondary school student teachers during a programming lab, where they investigate Riemann sums with programs in Python, and responses to a follow-up questionnaire on their experience during the lab.

Background

The connection between programming and learning in mathematics has been discussed since the late 1960s. During the 1970s, Seymour Papert developed the programming language LOGO with the aim of teaching children mathematics (Papert, 1980). The idea behind LOGO was that the visual language would make it easier to enable children to explore mathematical concepts. One advantage Papert could see in using programming in mathematics education was that it gave students an increased understanding of the problem-solving process.

There is a strong connection between programming and mathematics, partly because programming is based on logic, procedures, and functions (Misfeldt & Ejsing-Duun, 2015). Programming can help students gain insight into different mathematical concepts and develop mathematical abilities. For example, 12- to 13-year-old pupils developed understanding of the concept of variable using Logo-based tasks (Ursini-Legovich, 1994) and 6th grade pupils working with coding activities using Scratch showed significant improvement in mathematical thinking (Calao et al., 2015). Moreover, programming can be a good starting point for working with real world mathematical problems and connecting mathematics to applications in a new way, and thus has the potential to influence pupils' motivation and attitudes toward mathematics (Forsström & Kaufmann, 2018). In a study by Ke (2014), it was found that middle school pupils participating in Scratch-based mathematical game designing activities developed a significantly more positive attitude toward mathematics.

Nevertheless, it is not enough to know basic programming to be able to use programming in a mathematical context. In one study it was found that upper secondary school pupils with basic knowledge in Java programming experienced difficulties when using programming as a technical artifact to solve mathematical problems (Borg, 2021). These pupils had difficulties in translating mathematical ideas into programming code. In another study, in an activity where 5th grade pupils were expected to learn mathematical concepts through programming in Hopscotch, it turned out that the programming exercises themselves were not enough for the pupils to develop mathematical competences, but that with some help from the teacher, the pupils could understand how programming and mathematics were connected (Misfeldt & Ejsing-Duun, 2015). For teachers, the challenge of using programming to teach mathematics has multiple layers. It requires thinking through the programming aspects, the mathematical aspects, and the coordination between the two (Kilhamn, Rolandsson & Bråting, 2021).

An instrumental approach

Since the mathematics curricula for Swedish upper secondary school focus on programming as a tool, we use an *instrumental approach* (Trouche, 2004) to theoretically frame the study. This theory provides views concerning learning processes in complex technological environments. The key idea is the distinction between an artifact and an instrument: An artifact refers to a technological object that can be used as a tool, while an instrument is made up of the artifact together with a psychological component (Verillon & Rabardel, 1995). The artifact, such as for example a programming environment, becomes an instrument when the subject has integrated it into her activity. This process is called *instrumental genesis* and evolves in two directions: The *instrumentalization process* is directed towards the artifact itself, for example when a student learns how to handle the artifact, and the *instrumentation process* is directed towards the subject, for example when the student uses the artifact to represent a mathematical concept, changing the student's own understanding of the concept (Trouche, 2004).

An instrument involves the *techniques* and *mental schemes* that are developed and applied by the subject while using the artifact. The notion of a scheme was introduced by Piaget (1936) and redefined by Vergnaud as the "invariant organization of behavior for a given class of situations" (Vergnaud, 1996). Rabardel (1995) described a scheme organizing an activity with an artifact associated with the realization of a task as a *utilization scheme*. He distinguished between two types of utilization schemes: *usage schemes* that are oriented toward the management of the programming environment, and *instrumented action schemes* that are oriented to the carrying out of the specific mathematical task (Rabardel, 1995). To develop schemes and to build an instrument is a social process – it happens in a classroom environment with a collective process of instrumental genesis (Drijvers et al., 2010). Since the schemes cannot be observed directly, we can instead focus on the observable techniques, that is the interactions between the subject and the artifact.

Methodology

The participating students are secondary school mathematics student teachers taking their first semester of mathematics courses. The data collection was conducted about three quarters into the semester. The students were either in the end of their first or third year of the teacher education program, depending on whether mathematics is their primary or secondary subject. They had been introduced to programming in mathematics within their first mathematics courses on algebra and geometry. There they had built a block program in Scratch to draw regular polygons, using loops and variables. Thereafter, in the Calculus course, they are offered an optional workshop with an introduction to programming in Python, followed by a programming lab where they investigate how to approximate the area under a curve using narrower and narrower rectangles (Riemann sums) with Python. Participation in the lab is mandatory. In the workshop, the students were introduced to standard concepts of programming, such as

arithmetic operations, variables, and data types, before they were guided through some basic programming exercises involving for-loops, nested loops, if-else conditional expressions, the definition of functions, and some basic 2D graphic.

```
In [1]: def f(x):
            return x*x

In [2]: def RiemannTen():
            sum=0
            for i in range (0,10):
                sum=sum+(1/10)*f(i/10)
            print(sum)

In [3]: import matplotlib.pyplot as plt
        import numpy as np

In [4]: t=np.linspace(0,1,100)
        plt.plot(t,f(t))

        for i in range (0,10):
            x=np.array([i/10,i/10,(i+1)/10,(i+1)/10])
            y=np.array([0,f(i/10),f(i/10),0])
            plt.plot(x,y)
            plt.fill_between(x,y,facecolor='g', alpha=0.3)
        plt.title("Left Riemann sum for ten rectangles")
Out[4]: Text(0.5, 1.0, 'Left Riemann sum for ten rectangles')
```

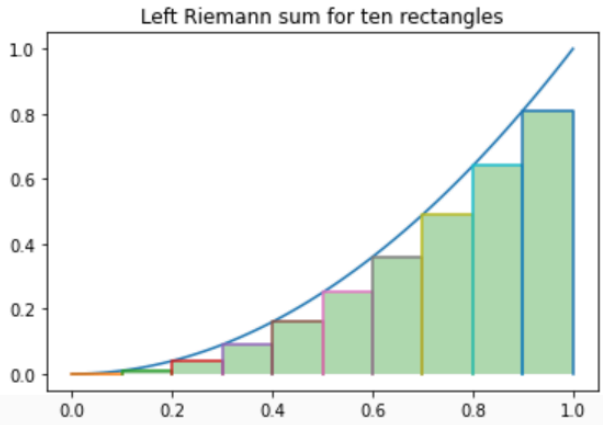

Figure 1: The code that the students received at the Python programming lab.

In the lab the students were reminded of the idea of a Riemann sum, which had been mentioned in a lecture, and were given some code: the definition of the function $f(x) = x^2$, the loop to compute the Riemann sum of this function from 0 to 1 with 10 intervals, and the code to visualize the graph of the function and the ten rectangles (see figure 1). The students were asked to make changes in it to investigate what happened if they changed the number of intervals when the Riemann sum was computed. Thereafter they were asked to investigate the difference between the left Riemann sum, the right Riemann sum, and the midpoint Riemann sum, and to see which one would converge the fastest. To do this they had to change the parameter in the definition of the Riemann

function (see ln [2] in figure 1) as well as the coordinates, i.e., the parameters in the two lists, in the loop that produces the graphic representation of the rectangles (see ln [4] in figure 1). Moreover, the students were asked to investigate Riemann sums for other functions, such as $f(x) = \sin(x)$ and $f(x) = e^x$, and other intervals $[0, B], B \in \mathbb{R}^+$. Finally, they were asked to compute, with the help of Riemann sums, the x-values of the quartiles of the function representing the standard normal distribution. At the lab the students followed the use-modify-create progression (Lee et al., 2011); they first had to copy and run the code, then to modify it and finally they – in the last exercise – had to write their own program. The students worked either alone or in pairs. At the end of the lab, they had to present their code and explain how it worked to one of the teachers in the course.

After the lab the students were asked to answer a questionnaire consisting of 12 questions: five multiple-choice and seven open-ended questions. The questions investigated their earlier experience of programming, their experience of the workshop and the two labs, and their understanding of programming in mathematics education in school and in teacher education. 38 students passed the lab and 26 of them answered the questionnaire. The data was collected in the spring of 2021, and due to the pandemic, the workshop and the lab were done on-line, using zoom as a conference environment. The questionnaire was also administered on-line. One multiple-choice question and two open-ended questions of the questionnaire focused on the earlier Scratch lab as well as students' expectations on the teacher education program and are not addressed in this paper. The remaining multiple-choice answers were analyzed statistically and the answers to open-ended questions were analyzed using qualitative content analysis. We looked for traces of the instrumentalization and instrumentation processes and focused on how students expressed the relationship between learning to program and learning mathematics, in general and in the context of this lab.

Results

We begin by presenting results concerning the students' experience of the lab and to what extent they found that it helped them in their understanding of Riemann sums. Thereafter we present their thoughts on programming in mathematics education.

Does the Python lab help the students in their understanding of Riemann sums?

The students have varying experience of programming when they start the teacher education program. Of the 26 students answering the questionnaire, a majority (14) claimed that they did not have any experience of programming, about one third (9) claimed to have some experience, and only three of the students claimed to have a lot of experience. The experienced students had knowledge of various programming languages such as C, C++, Java, JavaScript, Matlab and Visual Basic. Only four of them mentioned Python as one of the programming languages they had experience of.

All but two students choose to participate in the workshop. At the workshop the students were introduced to the programming environment and to the fundamental

concepts needed for the lab. All students appreciated the possibility to participate in the workshop, but due to lack of earlier experience some wanted more time, more basic examples, and easier exercises. At the lab we observed that some of the students were still in the beginning of their instrumental genesis; they were struggling with handling the concepts, structuring the program, using formal syntax, and debugging. They were frustrated when their program did not work as expected as they did not have a strategy for how to solve problems. It became more obvious that some students had difficulties understanding important programming concepts when they in the questionnaire were given a short program including a loop were the variable *sum* was computed through the successive adding of areas of rectangles (see ln [2] in figure 1) and asked to explain the meaning of the variable. Even though many of the students could give a satisfying explanation, showing that they could use such a program as an instrument, six of them were not able to do this. One of these students wrote: "I just accepted that it should look like this. I understand that the sum changes depending on the interval, but it looks very unclear to me". This student had after the lab possibly not yet developed sufficient usage schemes related to basic programming concepts; she could accept that the variable changed values in the loop but could not explain how and why it changed.

When the students were asked how difficult they experienced the lab to be, a majority (15) answered that it was difficult. Four of the students answered that it was easy, and the rest (7) answered that either the mathematics or the programming was difficult. In all, more than 80% of the students thought that the programming part of the lab was difficult. It is not surprising that all students with no experience of programming thought that the programming was difficult, and that all students with a lot of experience thought that the programming was easy (see Table 1).

Do you have experience of programming? / How did you experience the difficulty of the lab?	No experience of programming	Some experience of programming	A lot of experience of programming
It was easy	0	2	2
The programming was easy, but the mathematics was difficult	0	0	1
The mathematics was easy, but the programming was difficult	3	3	0
It was difficult	11	4	0

Table 1. The cross table shows the students' answers regarding their experience of programming and their experience of the difficulty of the programming lab.

One reason to include this programming lab in the analysis course was to support the students in using the experience of programming to get a deeper understanding of the concept of Riemann sums and the relation between integrals and the area under a curve, i.e., an instance of the instrumentation process. Asked whether the lab helped

them to understand Riemann sums, half of the students (13) answered yes, about a third (8) answered no and the remaining 5 were not sure. The students who answered 'yes' were asked to explain in what way the lab helped them. Their answers could be categorized into three types: *understanding the formula* (six students); *experimenting with parameters* (four students); *different representations* (three students).

The largest group answered that the lab helped them in *understanding the formula*. To write a correct program and make the intended changes, the students had to check each line of the program, in particular the parameters in the loop computing the sum. One student wrote: "I had to decompose the formula into smaller parts and that helped me understand how it was structured." A few students answered that they already understood Riemann sums, but that the lab clarified the formula for them.

Some students answered that *experimenting with parameters* helped them in their understanding. By changing the different parameters, they got an understanding for the role of each parameter in the program. One student expressed this in the following way: "I liked experimenting and changing the values. When I changed the number of rectangles, I could see how this affected the sum."

Finally, some students answered that using *different representations* helped them in their understanding. One of the students expressed that it is always easier to learn when the content is represented in different ways, and that she, after the lab, directly visualized the rectangles in the Riemann sum when she worked with integrals, demonstrating instrumentation. Another student wrote: "I could see the curve and how the rectangle changed depending on if it was left, right or midpoint Riemann sums."

Of the eight students who answered that the lab did not help them in their understanding of Riemann sums, all but one had no or little earlier experience of programming and thought that the programming part of the lab was difficult. It seems that these students were insecure mainly due to the programming since they had not developed sufficient usage schemes related to the management of the programming language; they were struggling, felt confused, and did not think that they were well enough prepared. The last student had a lot of experience with several different programming languages and thought that both the lab and the mathematics involved were easy. This student apparently already felt a sufficient understanding of the mathematics involved, and therefore didn't see the lab as leading to new knowledge. For this student, the instrumented action schemes related to the exercises in the lab were already developed.

Can programming offer pupils in school opportunities to learn mathematics?

To find out about the student teachers' understanding of programming in mathematics education, they were asked if they believed that programming can offer pupils in school opportunities to learn mathematics.

Out of the 26 students, 3 students answered 'no'. One student motivated this by writing that it takes too much time from the mathematics lessons. A second student motivated this through arguing that to be able to construct a program you already need

to know the mathematics involved and therefore the programming will not offer pupils the opportunity to learn mathematics. Thus, this student did not acknowledge the instrumentation process, i.e., she did not believe that programming can help students gain a deeper understanding of mathematical concepts. The third student had struggled a lot with both the mathematics and the programming part during the lab. Common to these three students was that neither of them experienced that the lab helped them in their own understanding of Riemann sums.

The remaining 23 students answered that they believed that programming can offer pupils in school opportunities to learn mathematics. They motivated this in several different ways:

- Programming can support a *deeper understanding* of mathematics, concepts, formulas, algorithms and so on (eleven students).
- *Visual representations* with the help of programming will help pupils in the learning of mathematics (four students).
- *Explorative work*, and changing the values of parameters in a program, will help pupils in the learning of mathematics (three students).
- Programming can support pupils in their *problem solving* (two students).
- Programming can help pupils to develop *logical thinking* (two students).
- Pupils are interested in gaming and robots, and they know that programming is important in future professions and therefore programming will be a *motivating factor* in their mathematics education (five students).

Out of the 23 students who were positive to programming in mathematics, nine also mentioned that more programming knowledge was needed. Teachers must learn programming, and how to teach programming in mathematics. Finally, some of the students mentioned that pupils need to work with programming from the early grades and throughout their mathematics courses. When the pupils have fundamental knowledge in programming and coding it will be easier to work with programming in a rewarding way and to offer pupils opportunities to learn mathematics.

Discussion

This study involved just one class-sized group of students and the specific observations may not represent teaching of programming in general. Rather, it provides a close-up view of how a session can attend to both programming and mathematical development. The students taking part in the study were generally positive regarding programming in mathematics, even though most of them had no, or only little, previous programming experience. It can be expected that they before the lab had not developed sufficient usage schemes related to fundamental computational concepts such as variables, parameters, and loops. Thus, most students had not come far in their instrumental genesis (Trouche, 2004) before the lab. The lab setup, applying a use-modify-create progression (Lee et al., 2011) where students were encouraged to help each other and solve difficulties together, offered opportunities for students to develop utilization schemes. In the instrumentalization process, they discovered the programming environment and

explored the basic computational concepts. The instrumentation process includes developing schemes for knowing when to perform an action, plan the action, and understand what the action does. During the lab, some students had difficulties reflecting on their work, but were pressed to do so in order to explain their code to each other or to one of the teachers.

Many students struggled with the mathematical content of the exercises during the lab. With a clearer guidance from the teachers, it may have been easier for the students to understand how the constructed program and the mathematical content were connected (Misfeldt & Ejsing-Duun, 2015). Moreover, many of the students did not have enough programming experience to be able to solve the exercises without support from other students or from the teachers. It can be difficult to construct exercises where programming is used to explore mathematics, or to solve mathematical problems, in such a way that neither the instrumental genesis, the programming nor the mathematics constitutes an overwhelming challenge for the students. However, this study shows that it is possible to attend to instrumental genesis and to mathematical learning in parallel.

A majority of the students in this study, even though they were only in the beginning of their instrumental genesis, recognized the potential of programming in mathematics teaching. Nevertheless, these future mathematics teachers face a complex challenge. They will be expected to establish conditions for pupils to acquire fundamental programming knowledge and to develop schemes related to the mathematical content at the same time. Here, the teacher education program has an additional responsibility: besides teaching programming and mathematics, we must also give student teachers opportunities to develop their competence in teaching programming in mathematics so that they, in turn, will be able to orchestrate their pupils' instrumental genesis.

References

Borg, A. (2021). *Designing for the incorporation of programming in mathematical education – Programming as an instrument for mathematical problem solving*. (Licentiate theses. Karlstad University). http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-85625

Calao, L. A., Moreno-León, J., Correa, H. E., & Robles, G. (2015). Developing mathematical thinking with Scratch: An experiment with 6th grade students. In D*esign for Teaching and Learning in a Networked World,* 1*0th European Conference on Technology Enhanced Learning, EC-TEL 2015* (pp. 17–27). Toledo, Spain: Springer. https://doi.org/10.1007/978-3-319-24258-3

Drijvers, P., Kieran, C., Mariotti, M. A., Ainley, J., Andresen, M., Chan, Y. C., Dana-Picard, T., Gueudet, G., Kidron, I., Leung, A. & Meagher, M. (2010). Integrating technology into mathematics education: Theoretical perspectives. In C., Hoyles, & J.-B., Lagrange (Eds.), *Mathematics education and technology – Rethinking the terrain* (pp. 89–132). New York: Springer.

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

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers & Education, 73,* 26–39. https://doi.org/10.1016/j.compedu.2013.12.010

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

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32–37. https://doi.org/10.1145/1929887.1929902

Misfeldt, M., & Ejsing-Duun, S. (2015). Learning mathematics through programming: An instrumental approach to potentials and pitfalls. In K. Krainer, & N. Vondrová, (Eds.) *Proceedings of the 9th Conference of the European Society for Research in Mathematics Education* (CERME 9) (pp. 2524–2530).

Misfeldt, M., Szabo, A., & Helenius, O. (2019). Surveying teachers' conception of programming as a mathematical topic following the implementation of a new mathematics curriculum. In U.T Jankvist, M. Van den Heuvel-Panhuizen, & M. Veldhuis (Eds.), *Proceedings of the Eleventh Congress of the European Society for Research in Mathematics Education,* CERME11, (pp. 2713–2720). Freudenthal Group & Freudenthal Institute, Utrecht University and ERME.

Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas*. Brighton: Harvester Press.

Pejlare, J. (2021). Programmering för lärande i matematik – beskrivning av ett forskningsprojekt. In: Y. Liljekvist, L. Björklund Boistrup, J. Häggström, L. Mattsson, O. Olande & H. Palmér (Eds.), *Sustainable mathematics education in a digitalized world. Proceedings of MADIF12.* (p. 265). SMDF.

Piaget, J. (1936). *La naissance de l'intelligence chez l'enfant*. Delachaux et Niestlé.

Rabardel, P. (1995). Les hommes et les technologies, approche cognitive des instruments contemporains, *Armand Colin*.

Skolverket. (2011). *Ämnesplan för matematik i gymnasieskolan 2011*. Revised 2021.

Trouche, L. (2004). Managing the Complexity of Human/Machine Interactions in Computerized Learning Environments: Guiding Students' Command Process through Instrumental Orchestrations. *International Journal of Computers for Mathematical Learning, 9*(3), 281–307. https://doi.org/10.1007/s10758-004-3468-5

Ursini-Legovich, S.; (1994). *Pupil's approaches to different characterizations of variable in Logo*. Doctoral thesis, Institute of Education, University of London. https://discovery.ucl.ac.uk/id/eprint/10021519/1/130463.pdf

Vergnaud, G. (1996). Au fond de l'apprentissage, la conceptualisation. In: R. Noirfalise & M.-J. Perrin (Eds.), *Actes de l'école d'été de didactique des mathématiques* (pp.174–185). IREM, Clermont-Ferrand.

Verillon, P., & Rabardel, P. (1995). Cognition and artifacts: A contribution to the study of though in relation to instrumented activity. *European Journal of Psychology of Education, 10*(1), 77–101.