Lesson Design and Evaluation of Programming in Mathematics Education

A Study of Experiences and Opinions about Using Programming for Teaching Integral Concepts in a High School Setting

Master's thesis in Learning and Leadership

VIDAR ERICSON
MOHAMMAD FUAD JAWEER

DEPARTMENT OF COMMUNICATION AND LEARNING IN SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022

MASTER'S THESIS 2022

Lesson Design and Evaluation of Programming in Mathematics Education

A Study of experiences and opinions about using programming for teaching concepts of integrals in a high school setting

Master's thesis in Learning and Leadership

VIDAR ERICSON
MOHAMMAD FUAD JAWEER

Department of Communication and Learning in Science
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022

Lesson Design and Evaluation of Programming in Mathematics Education
A Study of Experiences and Opinions about Using Programming for Teaching Concepts of Integrals in a High School Setting
VIDAR ERICSON, MOHAMMAD FUAD JAWEER

Supervisor:	Henrik Imberg, Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg
Co-supervisors:	Kristina Nilsson, NTI gymnasiet Johanneberg
	Mattias Börjesson, NTI gymnasiet Johanneberg
Examiner:	Philip Gerlee, Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg

Master's Thesis 2022
Department of Communication and Learning in Science
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Digitaltryck
Gothenburg, Sweden 2022

Lesson Design and Evaluation of Programming in Mathematics Education
A Study of Experiences and Opinions about Using Programming for Teaching Concepts of Integrals in a High school Setting
VIDAR ERICSON, MOHAMMAD FUAD JAWEER
Department of Communication and Learning in Science
Chalmers University of Technology

Abstract

Programming has been known to be an effective tool for mathematical problem solving since the advent of computers and programming. Dramatic developments in digital technology and computers as well as the fourth industrial revolution has led to a demand for employees with increased digital competence across varying fields, which has prompted educational institutions to adapt. Consequently, many countries in the EU were motivated to start using programming in mathematics teaching and therefore there is a need to know how programming can become an efficient part of mathematics education. This study investigates how programming can be integrated in mathematics to teach concepts of integrals. It was implemented by creating, conducting and evaluating two lessons where programming was used as a means for teaching students about two aspects of integrals, namely the fundamental theorem of calculus, and solids of revolution. Data was gathered through a survey among participating students and interviews with both students and teachers in conjunction with the eleven lessons that were held. It was found that in this context, programming can be valuable by *deepening understanding* and *facilitating work* while challenges arise in the form of *increased cognitive load, unclear purpose* and *varying knowledge.* Previous experience of programming correlated with finding the code easy to use and the programming enjoying, as well as a small positive correlation with learning something that would have been harder without programming.

Keywords: mathematics education, programming, teaching, integrals, solid of revolution, fundamental theorem of calculus.

Acknowledgements

First and foremost, we would like to thank our supervisor Henrik Imberg for his tireless support, guidance and interest in our project. His statistical analysis of the data and guidance regarding keeping an academic mindset and language were invaluable and gave many insights into the world of research. Without his support and encouragement, it would have been next to impossible to complete this thesis.

We would also like to thank Kristina Nilsson and Mattias Börjesson, our co-supervisors at NTI Johanneberg, where much of our work was conducted. Their experience and insight into the field of teaching helped us shape both lessons and surveys. At Hulebäcksgymnasiet we would like to thank Fredrik Nordling for organising the schedule for our time there.

Finally, we would like to thank all participating teachers and students for their valuable reflections and feedback.

Vidar Ericson and Mohammad Fuad Jaweer, Gothenburg, May 2022

Contents

List of Tables

List of Figures

1

Introduction

1.1 Background

In 1969, Feurzeig and Papert visualised a future where students of all ages learn the basic concepts of mathematics and methods through the newly developed programming language Logo, which they co-invented in 1967 (Eckert & Hjelte, 2021). In the 1980s, Seymour Papert, now a university lecturer and researcher at the Massachusetts Institute of Technology (MIT), argued that computational thinking ought to be used in mathematical education. Computational thinking can be defined as a set of problem-solving methods which explain a problem and its solution in such a way that a computer — human or machine — can effectively carry out the operations needed (Wing, 2014).

The idea of using *computational thinking* in mathematics education could not be widely applied in learning and teaching, because digital technology and tools had not been developed to the extent that they are today. In addition, digital tools were not available to general public as they currently are. The tremendous advances in computer science led to the idea being brought up again thirty years later. Jeannette Wing revived the term computational thinking in contemporary society, stressing that it should be taught in schools (Bråting et al., 2020).

In 2006, the European Parliament and the European Union proposed digital literacy as one of eight key competencies for lifelong learning (European Union, 2006). Based on that recommendation, most countries in the European Union started calling for the introduction of programming in school education. In addition, results from a review by the Swedish Schools Inspectorate (*Skolinspektionen*) in 2012 led to the introduction of digital competencies in curricula at both primary and high school. The Swedish Schools Inspectorate claimed that even though schools invest much money in buying digital equipment, not all teachers know how to use that equipment (Hellmark Knutsson & Nilsson, 2015).

On the other hand, the National Agency for Education (*Skolverket*) pointed out that the large difference in the use of *information technology* in the country's schools was considered so prominent that it infringes on everyone's right to an equal education.

They further argued that digital skills should therefore be included in the curricula in order to encourage all schools to teach digital competencies (Hellmark Knutsson & Nilsson, 2015).

In 2017, the Swedish National Agency for Education updated the Mathematics syllabus for all primary and high school education such that it includes programming in addition to traditional mathematical content. Since the start of the school year 2018-2019, these changes are supposed to be implemented in mathematics courses (Helenius et al., 2018), but difficulties have arisen since many teachers lack both knowledge and experience regarding programming (Jahnke, 2020; Stigberg & Stigberg, 2020). Due to this lack of knowledge and experience, there is a joint interest between teachers and the National Agency for Education to explore how programming can be integrated effectively into the mathematics curriculum.

ULF (Utbildning Lärande Forskning) is a research-based project that was established by the Swedish government to strengthen the scientific foundation that Swedish education is supposed to be built upon according to law (SFS 2010:800). This project emerged from the need for a stronger connection and increased cooperation between research and education. In this way, the government is attempting to broaden collaboration between universities researching educational sciences and the schools where education takes place (Wiberg, 2020).

NTI Johanneberg, a technology-oriented high-school in Gothenburg, has conducted several learning studies on the use of programming in mathematics education the past few years. Topics addressed include different representations of functions and integrals, with focus on how to use programming to teach and learn mathematical concepts. In contrast, the main focus of programming in mathematics according to the national curriculum is on problem solving (Skolverket, 2021). Results from previous studies at NTI Johanneberg indicate positive effects of programming also for learning mathematical concepts, but possibly mainly for students with a high level of prior knowledge in programming, and emphasises the need for further research in the topic (Lärteam matematik, NTI Johanneberg, 2020).

1.2 Purpose

The aim of this project is to study students' and teachers' experiences and opinions about using programming for teaching mathematical concepts of integrals in a high school setting through lesson design and evaluation.

1.3 Research Questions

The main question which this master thesis intends to answer is:

What are students' and teachers' experiences and opinions in applying programming as a means for teaching mathematical concepts about integrals?

This question is clarified by the following specific questions:

1. Which possibilities do teachers and students envision and experience while utilising programming as a means for teaching integrals?

2. What challenges do students and teachers anticipate and experience while utilising programming as a means for teaching integrals?

3. How does previous programming experience affect students' learning about integral concepts when programming is used as a means for teaching integrals?

1.4 Scope

The project's lessons are performed on Swedish high school students who study the course Mathematics 3c or Mathematics 4 in both the Science and Technology Programs. The lessons focus on integrals by dealing with either the Fundamental Theorem of Calculus or Solids of Revolution.

2 Theory

This chapter presents theories which were used in different parts of this project. Firstly, lesson design and the 5E model are presented as a starting point to design and plan two lessons to teach students about some particular integral concepts. Secondly, some aspects of learning theory are presented. Above all, Bandura's social learning theory and active learning were used as a basis for increased cooperation and learning when actualising lessons in the classroom. In addition, peer discussion is presented as a method to increase students' understanding. This method was frequently used during the lesson implementation to realise student activities. Finally, some knowledge of how programming was previously used in mathematics education and how it can be applied to learning situations is discussed. This was considered in order to avoid the errors that can occur when teachers attempt to combine mathematics and programming.

2.1 Lesson Design and the 5E Model

Kimberly Tanner (2010) presents one model which can be used to plan lessons. Tanner explains certain aspects which must be considered when the teacher plans the lesson. The model is called the *5E model* and was first introduced by Bybee et al. (2006). The model emphasises that the students should be interested in what is going on in the lesson. They must participate and be active during the lesson by discussing the new concepts which are displayed in the lesson and compare new knowledge with old to improve understanding of the material. Merrill (2002) presents the importance of teaching new knowledge by revising the old understandings. The author means that students ought to see how the new knowledge relates or changes the old one.

Tanner (2010) believes that students must have a chance to apply what they have learned during the lesson. Therefore, she recommends using the 5E model to facilitate lesson planning and students' learning. The model consists of five concepts: *Engage, Explore, Explain, Elaborate* and *Evaluate.*

The first E, Engage, means that the teacher creates a certain activity to capture students' attention, stimulate thinking and help them establish access to previous knowledge. The second E, Explore, aims to make students capable of exploring ideas and finding out what knowledge they already possess and what new thoughts can help them solve the tasks which are given during the lesson. For the third E, Explain, the teacher explains the new concepts to students which develops their previous knowledge to be able to accomplish the tasks. Under Elaboration, the teacher broadens the students' ability to use the new knowledge to solve exercises in additional contexts. For the last E, Evaluate, the teacher helps students to reflect on the lesson, to use the skills they have acquired and evaluate their understanding.

In this context, Namdar and Kucuk (2018) discovered that teachers found it difficult to find criticism of the 5E principle for planning the lesson, but they say:

- It is very difficult to find examples from daily life related to the subject.
- It is difficult for teachers to design new problem contexts for students to explore.

Consequently, it is arduous for teachers to plan their lessons according to 5E at all times (Namdar & Kucuk, 2018).

In any case, Tanner (2010) emphasises that when the teacher designs the lesson according to 5E concepts, he builds a class climate in which most students are active and pay attention. This results in a deep learning and understanding of the knowledge which the teacher wants to teach.

2.2 Learning Theory

Learning theories concern how learning takes place when people learn something new from parents, friends, teachers, leaders, school or society at large. These theories focus on what conditions and behaviours the teacher needs and should employ, along with what kind of roles the learners play in this process.

The theories presented in this section were chosen based on how influential they have been and previous experience regarding programming in mathematics education. In the authors' experience, students display a large variety in programming competence. Therefore, student collaboration was viewed as a central component of the lesson and theories regarding collaboration were prioritised. Additionally, theories concerning interactions between students and between students and teachers were prioritised in order to maintain a high degree of communication. This aims to help teachers evaluate student understanding and needs throughout the lesson.

2.2.1 Constructivism

Constructivism is one of the fields focusing on learning as changes in a person's mental schema. Jean Piaget was one of the most influential theorists in the subjects of pedagogy and epistemology. For several decades, Piaget's theory of children's developmental phases has influenced school systems around the world. Phillips and Soltis (2015) highlight that Jean Piaget considered human action to be a consequence of thinking and viewed the mind as a type of information processor. When information arrives to the brain, it is processed, assimilated and accommodated.

Constructivism focuses on inner mental activities and sees knowledge as a schema or symbolic mental construction. The schema is defined as "*a pattern of repetitive behaviour in which experiences are gradually internalised and coordinated*", (Phillips & Soltis, 2015). Coordinating boosts the schema and repetition results in a more powerful schema. Piaget claims that learning takes place when people take new things and add them to the schema, in other words, learning is a changing in the schema (Phillips & Soltis, 2015).

Tina Bruce, who also carried out research on schemas in the 1970s, considers schemas a type of cognitive psychology which studies the internal mental processes inside the brain. Thinking, attention, memory, language and learning are examples of mental processes which accompany the learning process and development of child. In addition, constructivism regards understanding as a necessary condition for learning. Moreover, mistakes are necessary for learning. They are an opportunity that, by overcoming, the knowledge that we consider correct is built. Finally, the society in which the individual lives has a great impact on their construction of knowledge (Phillips & Soltis, 2015).

2.2.2 Social Perspective on Learning

Social Learning theory takes into consideration the social contexts in which people are constantly learning. Lev S. Vygotsky, who is considered as the creator of the socio-cultural perspective, tried to describe human learning processes. He viewed learning as initiating a social activity when the learner acts in an interactive environment. Learning always occurs in the context of social groups, parents, brothers and sisters, friends and classmates with whom the learner can communicate and socialise. They can talk about things that worry them, engage in activities with their friends through language that is a social medium, read books and magazines. Therefore, social factors are very important in any learning process (Phillips & Soltis, 2015).

Learning new skills comes directly from experiencing the consequences of using that skill, or the process of observing others and seeing the consequences of their behaviour. Learning is also influenced by a person's self-efficacy, which is their judgement about whether they can successfully acquire knowledge and skills. Self-efficacy can be increased by verbal persuasion, logical verification, observation of others and past accomplishment (Phillips & Soltis, 2015).

2.2.3 Constructionism - Social Development Theory

Social development theory focuses on connections between people and also between a person and the social context in which they act and interact. Vygotsky points out that language is essential in the learning process. He considers speech, writing and media to be very important tools in any education (Phillips & Soltis, 2015).

Constructionism claims that development takes place during interactions between people and their surroundings. Vygotsky asserts that learners always have a zone which is called the Zone of Proximal Development (ZPD). This zone is situated between what the learner knows and understands now and how far he or she has the opportunity to go. There is always such a zone of opportunity which learners fill in different contexts by meeting other people with different perspectives and knowledge. It becomes challenging for people to move forward in this zone. Constructionism further asserts that learning is consciousness and cognition. Learning is the end product of socialisation (Phillips & Soltis, 2015).

2.2.4 Bandura's Social Learning Theory

Bandura's Social Learning Theory describes how people can learn something new by observing the behaviour of other people and applying rational mental behaviour. Albert Bandura considers observational learning to be the first step in social learning. Bandura's social theory is based on three main ideas (Bandura, 1969):

- Firstly, people learn through observing role models. Bandura identified three types of models in his experiment: a live model physically demonstrating an action, a live model using language to display a behaviour verbally, and a symbolic model showing behaviours in online media, movies, television programs and books.

- Secondly, internal psychology influences the learning process. This means that intrinsic reinforcements need to satisfy the psychological needs like a sense of accomplishment, satisfaction, a form of success and pride.

- Thirdly, learning a behaviour does not automatically mean that the person will execute it. Applying a new behaviour must be of value to the person for them to want to apply what they have learned.

There are four steps in Bandura's social learning Theory: attention, retention, reproduction and motivation.

- **Attention** - the learner needs to pay attention to the behaviour. People imitate behaviour that grabs their attention. Therefore, the more interesting the behaviours, the more fully the learner wants to engage with learning.

- **Retention** - how one can store the learned information. Bandura suggests a number of memory techniques such as repetition, writing it down and memory devices.

- **Reproduction**- the ability to perform the behaviour which was learned. It relies on the two first steps namely attention and retention. After attention and retention, the learner may move towards performing the observable behaviour. With further practice, the learner will undoubtedly improve and sharpen his or her skills. Practice makes perfect.

- **Motivation** - students need to be motivated enough to imitate the behaviour that was modelled (Bandura, 1969).

2.2.5 Active Learning

In a traditional class room, teaching occurs when a teacher stands in front of the students and gives a lecture while the students listen and take notes. The teacher asks questions and students answer. This method has been used in schools and universities all over the world for a long time. However, research has shown that traditional teaching has low learning efficiency. Many students hide themselves and the teacher can only observe the abilities of a few individual students. In addition, it does not enable students to use their knowledge in practice and most of them forget the knowledge which they gained shortly after the exam (Felder & Brent, 2016).

Before the Second World War, social and pedagogical researchers Allport and Watson suggested cooperative learning theory after finding that group work was more effective in terms of quantity, quality, and productivity in general than individual work (Gillies & Ashman, 2003). John Dewey expanded the cooperative learning principles by suggesting that students ought to apply the taught knowledge in everyday life. This is important since it helps students to find meaning in their learning by bridging the gap between theoretical and practical knowledge, leading to students better remembering the new information.

In their book, Felder and Brent (2016) claim that "*true learning results from doing things and reflecting on the outcomes, not from passively receiving information*". Felder and Brent (2016) present another method called Active Learning, which is based on cooperative learning. This method has shown successful results when it comes to students' long-term memory and deep learning. Active learning considers the learner to be the central component of teaching and not the teacher as in traditional learning. "*Active learning involves providing opportunities for students to meaningfully talk and listen, write, read and reflect on the content, ideas, issues, and concerns of an academic subject*" (Meyers & Jones, 1993).

Active learning is an alternative method to traditional learning. In active learning, the teacher gives students tasks where they ask each other questions, discuss problems and solve them together. This method helps students to link theory with things in everyday life. In addition, active learning is one of the best methods of making students capable of using their knowledge successfully in their future career. Moreover, active learning is especially well suited towards education in mathematics, engineering and technology, (Felder & Brent, 2016).

2.2.6 Peer Discussion

Michelle Smith et al. (2009) proved that discussion between classmates in the lesson creates a deeper understanding of what is being taught. In her study, students were asked to answer questions without discussion in groups. The answers were registered without alerting students to whether they were correct, and students were then asked to discuss the question in groups before answering again. The researchers note that the number of correct answers increased even in groups that did not have any correct answers at the first time (Smith et al., 2009).

Smith et al. (2009) demonstrated that discussion between classmates gives a deeper understanding despite the teacher believing that their explanation is much better than that of students. The teacher's explanation is often not enough to make all students understand the taught concepts.

In this context, Corrégé and Michinov (2021) argued that "*there is a lack of consensus about the effects of peer discussion on learning, and the issue about the optimal group size remains largely open to debate in this field*". However, the author claimed that small groups work much better than big groups. In big groups, every group member assigns the responsibility for discussion to others (Corrégé & Michinov, 2021). The authors suggest that groups ought to not contain more than three students.

2.3 Programming in Mathematics Education

H. Stigberg and S. Stigberg claim that one of the largest challenges of using programming in mathematics teaching in Swedish schools is that a large part of the teachers lack programming competence and experience *Skolverket* (hereafter shortened to NAE). In addition, they argue that the NAE needs to clarify the relationship between mathematics and programming. Most mathematics teachers are convinced that programming is effective in developing students' problem-solving skills, but with the new curriculum the NAE in Sweden will use programming to develop logical thinking and not just problem solving (Stigberg & Stigberg, 2020).

Research is not conclusive on whether programming can develop-problem solving or other mathematical skills. However, it is necessary to know how programming can consolidate mathematical subjects in different mathematics courses, because programming has been integrated into all mathematics syllabi (Stigberg & Stigberg, 2020).

Furthermore, Stigberg and Stigberg (2020) think that the NAE ought to create a mathematical curriculum which is rich in programming materials. Teachers need more resources detailing how to use programming in mathematics education. One-day workshops do not provide teachers with sufficient knowledge and experience to use programming in mathematics teaching (Ball et al., 2008). In addition, a strong motivation is needed to show teachers and students why programming is relevant to use in mathematics education. Otherwise, the process will not be serious and meaningful (Stigberg & Stigberg, 2020).

Forsström and Kaufmann (2018) argued that all studies which support using programming in mathematics education were based on three main themes: *the motivation to learn mathematics, student performance in mathematics, and the collaboration between students which results in a changed role for the teacher.* The authors show that in certain circumstances, programming increases students' motivation to learn mathematics and their participation during mathematics lessons (Forsström & Kaufmann, 2018).

However, Forsström and Kaufmann (2018) argued that the relationships between students, and teacher and students look different when programming is used in mathematics teaching. The authors focus on the need for collaboration in the classroom, which should be taken care of during the lesson design phase (Forsström & Kaufmann, 2018).

Eckert and Hjelte (2021) argue that programming and mathematics can take different positions in relation to each other. Programming can be used to learn mathematics and mathematics can be used to learn programming. Therefore, the teacher ought to be careful in choosing:

1. Mathematical learning objectives for the task - what the teacher thinks the students should learn when working on the task,
2. An assignment presented to the student, either a mathematics assignment or a programming assignment and
3. Tools the student uses to solve the task (Eckert & Hjelte, 2021).

Finally, Anette Jahnke (2020) demonstrates important perceived advantages of applying programming in mathematics courses in Sweden:

- Teachers' collaboration was strengthened,
- Lesson study strengthened the quality of collaboration,
- Teachers developed new and collective knowledge about programming,
- Increased knowledge of how programming can be used,
- Dissemination of knowledge through conferences and research articles,
- Students became more positive to programming instruction.

In addition, Jahnke (2020) calls attention to essential challenges when using programming in mathematics education:

- Limited knowledge among teachers
- Difficult for teachers with limited knowledge to create lesson plans that can also improve mathematical abilities
- Progression unclear in syllabus between levels in primary school.

3 Method

Two programming-based lessons were developed, implemented and evaluated during this project. The chapter begins by presenting both created lessons, followed by a description of the study population. Thereafter, the design and analysis of the student survey is discussed. Finally, the design and analysis of the interviews is discussed.

3.1 Lessons

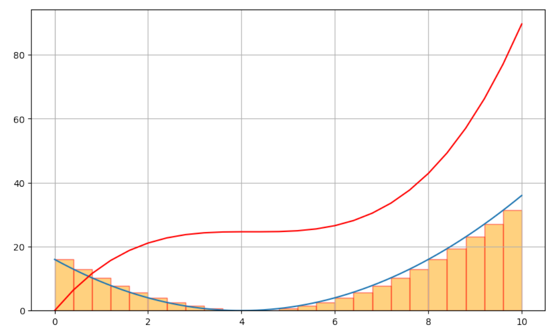
Two different lessons were created, the first with a focus on the fundamental theorem of calculus (later referred to as **Lesson 1**), and the second with a focus on solids of revolution (later referred to as **Lesson 2**). Both lessons were created in collaboration with a team of high-school mathematics teachers at NTI Johanneberg. Preliminary versions were discussed and developed in accordance with their feedback. For the complete lesson material presented to students, see the GitHub repository found on https://github.com/evidar/thesis_lessons. A summary of both lessons as well as their lesson plans are presented below.

The primary challenge in creating the lessons was in making them available to students of very varying programming knowledge. In order to work toward this goal, most of the work that the students were assigned to involved manipulating already finished code instead of creating their own code.

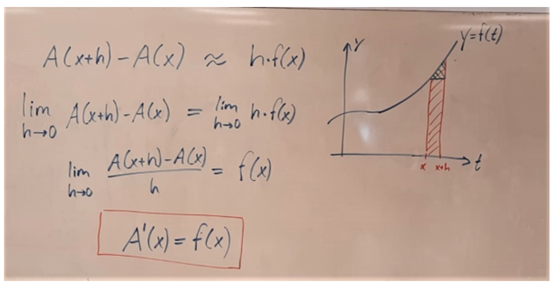
In order to keep data collection as reliable as possible, attempts were made to keep the lessons similar between different occasions. A time plan was created and lessons were practised between the two members of this thesis to form consensus regarding the overall execution and what parts to emphasise. For practical reasons and due to difference in schedule between classes, the lessons varied in length from 70 to 90 minutes.

3.1.1 Fundamental Theorem of Calculus

Lesson 1 was based on a lesson on the fundamental theorem of calculus previously developed, implemented and tested at NTI Johanneberg. The lesson was created to fit the material in the course *Mathematics 3c*, where students for the first time encounter derivatives and integrals. Students were expected to possess basic prior knowledge of Riemann sums, primitive functions and the concept of integrals as an area under the graph of a function (Table 3.1 and Figure 3.1).



(a)


$A(x+h)-A(x) \approx h\cdot f(x)$
$\lim_{h\to 0} A(x+h)-A(x) = \lim_{h\to 0} h\cdot f(x)$
$\lim_{h\to 0} \frac{A(x+h)-A(x)}{h} = f(x)$
$\boxed{A'(x)=f(x)}$
y
y=f(t)
x
x+h
t

(b)

Figure 3.1: Figures from Lesson 1. Figure **(a)**illustrates a visualisation of Riemann sum, area function and the integrand. Figure (**b**) illustrates the algebraic proof shown at the end of the lesson. The graph of the integrand $f(x)$ is the blue line in **(a)** while the graph of the area function, $A(x)$, is the red line. A(x) represents the area function. Note that the functions graphed in **(a)** and **(b)** are not the same.

Lesson Description

Aim: To understand the *Fundamental Theorem of Calculus* and the connection between Riemann sums, integrals and primitive functions.

Outline:

- Introduction of lesson aim and structure.
- Students work in pairs to calculate Riemann sum of one function in a handout.
- The teacher presents code that calculates the Riemann sum. Students think about the code and discuss it with their programming partner. The teacher summarises and changes the value of parameters to show students how they affect the result.
- Students use the code to do the same thing which they have done manually in the handout, but with the option of increasing the number of rectangles in the Riemann sum. This illustrates the benifits of programming. Students discuss how the code works and what it does. A few minutes are devoted for students to manipulate the code.
- The teacher presents the *area function*, what it means and how it is calculated using the Riemann sum (Area function $A(x)$: is a function to calculate area between integrand and x-axis in the range $[0, x]$).
- Students experiment with how accumulated area varies depending on function.

Students discuss the relation between area function and integrand. The teacher summarises discussion, concluding that the area function appears to be the same as the primitive function. The idea is reinforced by fitting a polynomial to the values $A(x)$ of the area function.

- Algebraic proof of the fundamental theorem of calculus.
- Teacher and students summarise the lesson.
- Students answer the survey.

Table 3.1: Lesson plan on fundamental theorem of calculus (Lesson 1)

Element	Content	Purpose	Time
1. Introduction	Explanation of the format of the lesson. Repetition of Riemann sums, primitive functions and the concept of integrals.	Evoke thoughts from previous lessons.	5
2. Coding			25
2.1 Estimate of Riemann sum	Students receive a handout with the graph of a polynomial and are tasked with estimating the value of the definite integral by hand.	Doing it by hand may increase understanding of the program.	
2.2 Program specification	Students think about what the program is supposed to do and receive the following questions: • What should the program be able to do? • Which input will it require? • Which output will it produce?	Increase understanding of the program that will be presented later.	
2.3 Summary	The teacher leads a summary of 2.2 and presents the structure for the code.		
2.4 Share code	Students receive code and discuss how it works. Thereafter students are tasked with modifying the code so that it solves the problem on the handout in 2.1 but with a more precise answer.	Students understand the purpose of using programming in mathematics.	
3. Area Function			18
3. The Area Function	The teacher motivates the existence of the area function and explains its properties.	It should be explained since it is not obvious for everyone that there exists an area function.	
3.2 Plotting	Students are presented with code that plots coordinates in a graph. They use their program from 2.4 to plot x-values alongside each value's accumulated area.	Connection to the handout, students see how programming can be valuable.	
3.3 Conclusions	From the graphs, students attempt to form conclusions about how the area function is may be related to the base function.	Challenge students thinking through experimental work. Attempting to first make the connection themselves increases engagement in the task.	
3.4 Regression	The teacher demonstrates that the area function appears to be the primitive function of the base function through regression of the plotted points. The teacher further explains what this means: that the derivative of the area function is the base function.	Seeing that their hypothesis is correct may increase engagement in the upcoming proof.	
4. Proof			10
4.1 $A(b) - A(a)$	Our program can only calculate the area between 0 and x. What can we do to calculate it between 3 and 4? Discuss.	Broaden the concept.	
4.2 Growth of Area Function	How does the area function increase? One way of calculating this is through the area between 2 and 2.1. Students use their programs to do this.	If students are engaged and contributing in the proof their understanding of it may increase.	
4.3 Alternative Growth	The teacher shows students a different method for approximating the increase in an interval - creating a rectangle with width h and calculating $h \cdot f(x)$. This method is motivated intuitively by an illustration and shown to give the same answer as the previous method.	A step in understanding the proof, where numerical and abstract aspects are connected.	
4.4 Proof	The teacher now writes the equation in its general form, $A(x+h) - A(x) = h \cdot f(x)$, divides by h and takes the limit where $\lim h \to 0$. We have now proven that the derivative of the area function and the base function are the same thing.	Connection between lesson activities and the formal mathematical proof, making abstract mathematics more understandable	
5. Summary	The teacher and students summarise what they have discovered. It is important to remark that all three representations are of the same thing. The limit of the Riemann sum = the integral between a and b = $F(b) - F(a)$. Students then answer the **survey**.	Tie the lesson together and make conclusions common to everyone in the classroom.	12

3.1.2 Solids of Revolution

In accordance with the decided scope, the second lesson also had to cover some aspect of integrals. After examining the curriculum and mathematics textbook of the course *Mathematics 4*, some potential subjects were defined. Thereafter, a discussion with experienced teachers was held to explore which area students could benefit most of an additional, programming-based, lesson. This resulted in choosing *solids of revolution* as the subject for Lesson 2. Preliminary versions of the lesson were discussed with a group of high school teachers in mathematics and developed according to their feedback (Table 3.2 and Figure 3.2).

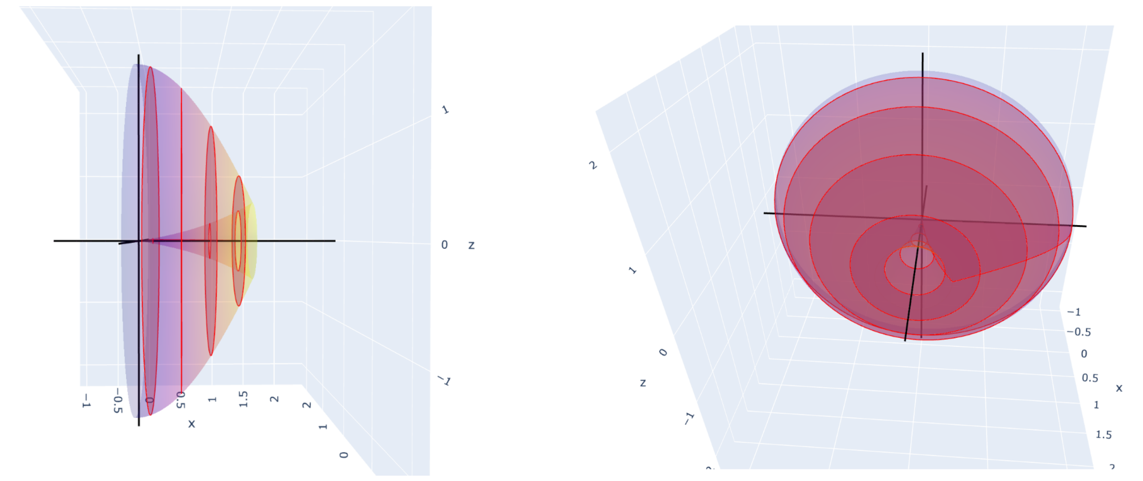


Figure 3.2: Example of the visualisation of solids of revolution that Lesson 2 provides. The figure shows the solid that is created between the two functions $f(x) = 2\cos(x)$ and $g(x) = \frac{e^x - 1}{2\pi}$ when they rotate around the x-axis for $x \in [0, 1.345]$, from two different perspectives. Four cross-sections of the solid are also illustrated. Interactive visualisation software were used that enabled students to rotate and zoom in or out in the figures.

Lesson Description

Aim: increased understanding of solids of revolution. Understand the formula for calculating the volume obtained by two functions that are rotated around around a coordinate axis.

Outline:

- The teacher presents code which draws axes in a three-dimensional coordinate system. Students reflect, discuss and experiment with the code to find out how the code work.
- The teacher presents code to draw a function in the $x - y$ plane of a three dimensional coordinate system. Students are tasked with creating a new function and manipulating the code and the 3D-graph.

- Students work in pairs to answer how the area between two functions can be calculated. If time permits, algebraically calculate an integral between two functions of their choice. Thereafter, the question is discussed in the entire class and summarised by the teacher.
- The lesson moves from 2D to 3D and drawing solids of revolution. Students discuss what a figure looks like when a graph of a constant and a linear function is rotated around the x-axis.
- The teacher presents the code that generates the solid of revolution of one function. Students discuss the code, change the mathematical function and parameters to get different 3D-graphs.
- The teacher present code that generates the solid of revolution created from rotating the region between two functions. Students discuss the code, change the mathematical functions and modify the code to get different 3D-graphs
- The teacher explains the concept of a **cross section** and shows a piece of code that draws a cross section. Students discuss the cross section and the related piece of code, change the mathematical functions and modify the code to illustrate one cross section.
- The teacher presents a piece of code that draws draw a multiple cross sections. Students experiment with how cross sections are shown.
- Students work in pairs to determine a formula for the area of each cross section and attempt to generalise it to arbitrary functions. Thereafter, the question is discussed in the entire class and summarised by the teacher.
- The teacher summarises how the area of an arbitrary cross section can be calculated, deriving the formula. The teacher shows students where they should write the formula in the code to calculate cross section area. Students experiment to get the area of a cross section between a pair of functions of their choice.
- Students work in pairs to answer: *how can the value of the volume of rotation be determined by the area of the cross section?* Thereafter, the question is discussed in the entire class and summarised by the teacher.
- The teacher summarises how the value of a volume can be calculated via integrating area of the cross-section. The teacher writes a piece of code to calculate the integral and students experiment with functions of their choice.
- The teacher and students repeat and summarise the lesson.
- Students answer the survey.

Table 3.2: Lesson plan on solids of revolution (Lesson 2)

Element	Content	Purpose	Time
1. Introduction	Introduction of teachers and the project that the lesson is a part of. The teacher provides an overview of the lesson structure - programming will be done in pairs, lots of discussion and a survey at the end.	Capture students' interest and make them comfortable in participating in the lesson. Introduce the structure of the lesson.	3
2. Solids of Revolution			54
2.1 Repetition	Question: What is a solid of revolution? Students discuss and the teacher summarises.	Refresh knowledge and prime students for learning more about solids of revolution.	3
2.2 Introduction to Code	Summary of what the code will be able to do at the end of the lesson. ***Task:** what does the code do? Discuss with your programming partner.* The teacher summarises discussion.	Demystify code, build familiarity with the code base which later programs are built upon.	5
2.3 Integrals in 2D	Students are presented with code that draws the integral of a function in the xy-plane. ***Task:** Change the functions to those presented.* The teacher presents code which graphs two functions and the area between these. ***Task:** Define a new function and draw the area between your two functions.* ***Task:** How can you determine the integral between these two functions algebraically? Discuss with your partner.* Code-along of realisation in code.	Make students comfortable with manipulating programming through familiar mathematics. Building understanding of programming functions for future use.	13
2.4 Visualise solids of revolution	Students attempt to visualise some solids of revolution with pen and paper. ***Task:** What do the following solids of revolution look like?*	Get students thinking about properties of solids of revolution. Emphasise difficulty of visualising 3D figures without digital tools.	3
2.5 Draw Solids of Revolution	The teacher presents code with which students can draw the solid of revolution for a given function. ***Task:** Change the function being rotated to those introduced earlier or to an arbitrary one.* Thereafter, code is presented to draw the solid that is created from rotating the area between two functions. ***Task:** experiment with different functions, either from a list of suggestions or arbitrarily chosen ones.*	Assist students in visualising solids of revolution. Students learn to easily experiment with what function is used and how that affects the solid of revolution.	8
2.6 Cross sections	Code is presented which draws a cross section at a specific value or multiple cross sections of the solid of revolution. ***Task:** Determine a formula for the area of a cross section. Attempt to generalise it to arbitrarily chosen functions.* The teacher summarises the solution through an illustration.	Important building block for students to later understand the formula for volume of a solid of revolution between two functions.	8
2.7 Volume of a Solid of Revolution	***Task:** How can the volume of the solid of revolution be determined through knowledge of the cross section's area? Discuss.* The teacher summarises on the whiteboard with the help of an illustration and writes the mathematical formula. ***Task:** Complete $area\_func(x)$ and replace volume in order to be able to calculate the volume with the created program.*	Increase student understanding of the formula for volume of a solid of revolution and what its components represent.	14
3. Summary	Discussion in pairs and together about what the lesson was about, which mathematics was involved and how programming could be useful. Students answer the **survey**.	Summarising and repeating knowledge so that students better remember the lesson.	15

3.1.3 Programming Language and Environment

Both lessons were implemented and taught using the ***Python*** programming language (Van Rossum & Drake, 2009) and the Google Colaboratory programming environment (Google Colab, 2022). 3D-plots used for illustration and teaching solids of revolution (Lesson 2) were implemented using Plotly version 5.5.0 (Plotly, 2022).

Python is easy to learn and have been using in many fields of education because it offers an interactive environment in which students can easy explore and learn functions, procedures, data structure and syntax. Python has a large numbers of libraries which make it very suitable to cover and perform most of programming applications, (Chen & Liu, 2022).

In addition, Chen and Liu (2022) says that "*the Python language abandons complex syntax and chooses one that is clear and rarely ambiguous*". Moreover, simplicity of Python helps the learner to focus on the problem instead of focus on programming language, it facilitates the learning process. Finally, Python is considered as high-level language which can be used without thinking in underlying details (Chen & Liu, 2022).

Google Colaboratory or "Colab" for short, is used as a programming environment. It is a very simple platform which allows any Google user to write and execute Python code and is especially well suited to education, data analyses and machine learning, (Tock, 2019). Google Colab facilitates students work to write the code and to bring ready programmes from the Google Drive. Moreover, it allows the teacher to hide certain parts of the code which students do not need to see.

3.2 Study Population

Eight classes from two different schools were included in the study. Five of the classes participated in one programming lesson, while three of the classes from one of the school participated in both lessons. Thus, a total of eleven lessons were held. Two classes attended the second year of the Science Program and were presented only with Lesson 1, three classes attended the third year of the Science Program and were presented only with Lesson 2, and three classes attended the Technical Program and were attended both lessons. The three Technical Classes were studying the course *Mathematics 3c* were solids of revolution are not normally an element but they had some lessons devoted to this as advanced studies before Lesson 2 was presented.

The Science students did not have programming as an ordinary compulsory part of their education, although a few students had chosen programming as one of their elective courses. Most of the Science students had only encountered programming within isolated lessons in mathematics during high school, along with some varied knowledge from their earlier education. The Science students had finished and already been tested on their knowledge of integrals, whereas the Technology students were still studying this chapter.

All of the Technology students had at least started a programming course based in the *Ruby* programming language, which is similar to *Python*. Students from the Technical high school were familiar with programming and had some experience of writing their own code for problem solving.

In addition, eleven teacher-interviews were conducted with six different mathematics teachers in both Technical and Science schools. Some teachers have been interviewed more than once because they teach multiple classes.

3.3 Survey Design and Analysis

Survey questions were created based on the previously defined research questions in section 1.3. These questions were discussed and improved with focusing on getting answers to all relevant questions while keeping the survey short to get a high response rate, a methodology endorsed by Esaiasson et al. (2017). As a part of keeping the response rate high, only the rating questions were made obligatory while the longer free-text questions were not.

Generally, Rating questions had the same scale of *1 - strongly disagree, 2 - disagree, 3 - neutral, 4 - agree, 5 - strongly agree.* Questions were also posed in such a way that giving a high rating equates to having a positive impression of the lesson. To the extent possible questions were posed in a neutral way.

The survey contained three different types of questions:

1. Previous programming experiences;
2. Lesson implementation and understanding of the material;
3. Outcome in the form of increased understanding of concepts and the benefits of using programming for this purpose.

The survey was concluded by asking students if they wanted to participate in a short interview. A full list of the survey questions can be found in Appendix A.

3.3.1 Statistical Methods

Data are presented descriptively as numbers and percentages of survey answers, and analysed using non-parametric statistical methods. Comparisons between groups were performed using Wilcoxon rank sum or van Elteren test (i.e., stratified Wilcoxon test) (Wild, 1997). Associations between responses to different survey items were analysed using Cochran-Maentel-Haenzel test, and the strength of association summarised using Spearman partial correlation coefficient (Agresti, 2003). The partial Spearman correlation coefficient describes the strength of monotone association between two variables x and y, while controlling for one or more background variables, as a real number $r \in [-1, 1]$ (Artusi et al., 2002). A coefficient $r > 0$ corresponds to a positive association (increase in x associated with increase in y), $r < 0$ to a negative association (increase in x associated with decrease in y), and $r = 0$ to no (monotone) association (Artusi et al., 2002). All tests with $p < 0.05$ were considered statistically significant, where the p-value is the probability to obtain a result that is at least as extreme as the observed outcome given that no association exists.

Statistical comparisons were stratified on school, lesson, or school and lesson, as appropriate. Stratification was used to ensure that comparisons were made between similar groups of students, e.g., to account for differences between lessons when comparing schools, or account for differences between schools and lessons when analysing correlations between survey items. Further, stratification can be used as a non-parametric method to account for repeated measurements on the same subjects, which in our case was present since three of the classes participated in both lessons.

Statistical analyses were performed using `R` software for statistical computing, version 4.1.0 (Venebles & Smith, 2022). Non-parametric tests were performed using the `coin` package, version 1.4-2 (Hothorn et al., 2021), and partial Spearman correlation coefficients calculated using the `ppcor` package, version 1.1 (Kim, 2015).

3.3.2 Analysis of Free-Text Answers

In order to better analyse the Free-text answers to survey question that students gave, these were categorised into groups with similar answers. To decide which categories were relevant and interesting, both authors started looking through answers in order to get an idea of what they were about. Thereafter potential categories were discussed until these were decided, followed by each author going through half of the answers and categorising them, marking any answers that were difficult to group. Potential new categories were discussed and each difficult answer was discussed, either placed in a category or marked and discarded if they did not fit into any categories and only one or very few students expressed the same sentiment.

3.4 Interview Design and Analysis

Questions for the interviews were constructed to be neutral and ordered in such a way that general questions were asked before more specific ones. The purpose of this was both to get an idea of what left strong impressions and to lessen interviewer effects such as the respondent being affected by the interviewer's prejudices. Additionally all interviews started with some easy-to-answer warm-up questions about the student's previous knowledge and experience in programming. Both of these methods are espoused in Essiasson et al *Metodpraktikan* (2017).

3.4.1 Student Interviews

Among the students that were willing to participate in an interview, two were randomly chosen after each lesson. The goal was for one student to have a relatively large amount of programming experience, and the other one to have a relatively small amount of experience. However, since only a few student volunteered, sometimes both students had relatively high programming experience. For all interviews, the instructor who had held the lesson was responsible for taking notes while the other instructor was responsible for conducting the interview.

At the start of the interviews the privacy of opinions that students voiced was made clear - teachers and peers would not know what had been said during the interview while any quotations used would be anonymous. During the interviews students were actively encouraged to name areas of improvement or bad aspects of the lesson emphasising that negative opinions are valuable. This was done since there may otherwise be a risk of students mostly discussing good things about the lesson in order to avoid uncomfortable situations.

3.4.2 Teacher Interviews

Teacher interviews mostly followed the same principles as student interviews, starting with broad questions and gradually narrowing them down. These interviews were generally longer than student interviews and teachers mostly had more feedback to give than students. In addition to questions relating to their own experience of the lesson, teachers were also asked in general about classroom climate and their perception of student understanding and attitudes towards the lesson.

Teachers were also asked about the possibilities of programming in mathematics class, their own experience and the difficulties of working with programming in mathematics for teachers and students alike.

3.4.3 Thematic Analysis

In order to condense the immense amounts of data that interviews with students and teachers gave, a thematic analysis of the interviews was conducted around the two central phenomena ***benefits of using programming in mathematics*** and ***difficulties existing when introducing programming into mathematics***. In a thematic analysis, the researcher attempts to identify all aspects of a phenomenon, see *Metodpraktikan* (Esaiasson et al., 2017), page 281-282 for a short summary of this thematic analysis. Note that the project did not undertake a full thematic analysis. In contrast to a full thematic analysis, interviews were not fully transcribed but summarised in notes twice: once during the initial note taking of the interview and then a second time while listening through all audio recordings of the interviews. Interviews regarding different lessons were combined because both lessons were examples of using programming to teach some aspect of integrals, but with different implementations.

4 Results

This chapter presents the findings of the study by summarising data gathered through surveys and interviews. In general, subjects are discussed in the same order as related questions were asked on the survey. After presenting Table 4.1 and Table 4.2 which summarise survey answers, the chapter continues with *Previous Programming Experience* (Section 4.1) and *Understanding of the Lesson* (Section 4.2). Thereafter, *Difficulties During the Lesson* (Section 4.4) and *Possibilities of Programming in Mathematics* (Section 4.5) are presented, finally ending in *Lesson Effect on Learning* (Section 4.3).

4.1 Previous Programming Experience

Results which are presented in Table 4.1 show that Technology students had much better previous programming experience than Science students. Table 4.1 shows that 44% of Technology students versus 89% of Science students claimed that they had not programmed in their leisure time. Survey results showed that 97% of Technology students versus 17% of Science students (Table 4.1) claimed that they had previous experience of programming through studying certain programming courses in the school.

4.1.1 Sufficient Programming Knowledge to Understand

Results from Table 4.1 showed that only 67% of Technology students versus 30% of Science students considered themselves to have enough knowledge in programming before the lessons. Consequently, 30% of the Technology students, who had studied a course in programming at school, thought that they did not have enough programming knowledge for the lesson. On the other hand, at least 13% of Science students, who think that they do not have a good previous programming knowledge in the school, say that they have enough programming knowledge to understand the lesson.

4.1.2 Correlates of Previous Programming Experience

There was a strong positive correlation ($r = 0.54, p < 0.0001$) between "*enough prior knowledge in programming*" and "*the code was easy to use and understand*", see

Figure 4.1. In addition, four moderate positive correlations were noticed between different categories. There was a moderate positive correlation between "*enough prior knowledge in programming*" and "*fun to use programming*" ($r = 0.35, p < 0.0001$), "*previous programming in school*" and "*the code was easy to understand*" ($r = 0.19, p = 0.011$), "*programming on leisure time*" and "*the code was easy to understand*" ($r = 0.31, p = 0.0003$) and between "*programming on leisure time*" and "*fun to use programming*" ($r = 0.28, p = 0.0006$).

4.2 Understanding of the Lesson

4.2.1 Code and Mathematics were Easy

Table 4.1 shows that 64% of Technology students versus 43% of Science students agreed with that the code was easy to use and understand. In addition, 70% of Technology students versus 75% of Science students (see Table 4.1) agreed that the mathematical content was easy to follow.

From Figure 4.1 some correlations between high rankings in different statements were observed. High rating of both the statement "*Code was easy to use and understand*" and "*Mathematics was easy*" have a moderate correlation to "The lesson increased understanding of integrals" ($r = 0.31$ and $r = 0.27$ respectively, both with $p < 0.0001$).

The aspect of having to understand both the mathematics and the code is also something that students had brought up in the interviews:

> *I understood what it [the code] did and it was good for me that I was comfortable with it, because that helped me to focus more on the maths when I understood the code anyway.*

4.2.2 Code was Fun

Slightly less than half of Technology students and Science students (48% and 42% respectively) considered the programming used during the lesson to be fun (see Table 4.1). Moreover, Figure 4.1 shows moderate correlation ($r = 0.46; p < 0.0001$) between "*The code was easy to use and understand*" and "*It was fun to use programming in the class*".

4.2.3 Strengths and Weaknesses of the Lesson Plan

Exactly half of all answers in Lesson 1 and 62% of all answers in Lesson 2 considered the structure of the lesson to be good. In addition, 9% and 14% of answers in Lesson 1 and Lesson 2 respectively claimed that the experimentation that the lesson provided was one of its primary strengths, see Table 4.2.

Regarding weaknesses, 39% of answers in Lesson 1 and 49% of answers in Lesson

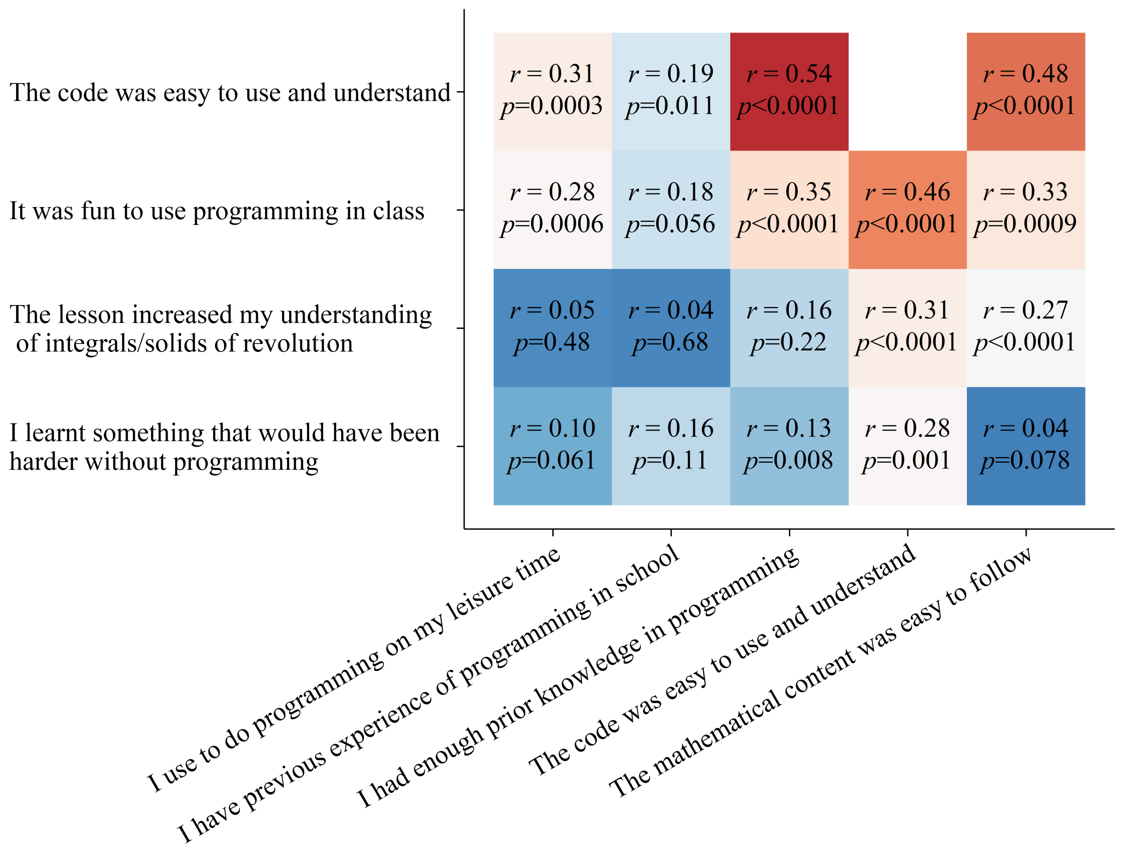


Figure 4.1: Correlations between student ratings on statements about the lesson. Dark blue represent no correlation, whereas a more intense red colour represents a stronger positive correlation. A correlation coefficient $r > 0$ corresponds to a positive association (increase in x associated with increase in y), $r < 0$ to a negative association (increase in x associated with decrease in y), and $r = 0$ to no (monotone) association. All tests with $p < 0.05$ are considered statistically significant, where the p-value is the probability to obtain a result that is as least as extreme as the observed outcome given that no association exists.

2 stressed that it would be appropriate to have more time. Furthermore, 14% of answers in Lesson 1 and 18% in Lesson 2 considered the lesson to pose too little of a challenge to students (Table 4.2).

4.3 Lesson Effect on Learning

4.3.1 Effect on Understanding of Mathematics

More students in Lesson 1 agreed than disagreed with the statement "the lesson increased my understanding of integrals" (48% versus 23% and 38% versus 20% for Technical and Science Schools, respectively). However, a large amount of students did not gain an increased understanding (Table 4.1). 33% of total answers felt that they already knew most of the mathematics that was presented in the lesson which

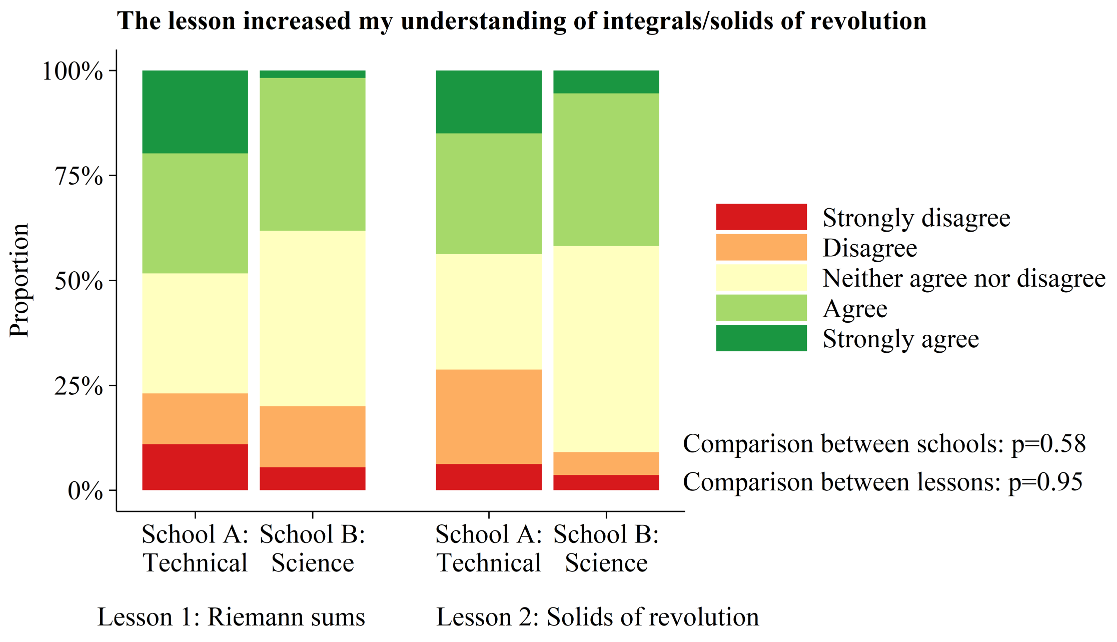


Figure 4.2: Perceived effect of the lesson on student understanding of integrals (Lesson 1) or solids of revolution (Lesson 2). The data in this figure is also found in Table 4.1 where some categories were merged. All tests with $p < 0.05$ are considered statistically significant, where the p-value is the probability to obtain a result that is as least as extreme as the observed outcome given that no difference exists.

was also the primary reason as to why they felt that the lesson did not increase their understanding (Table 4.2).

The situation in Lesson 2 resembled that of Lesson 1. The proportion of students who agreed versus disagreed was 44% versus 29% and 42% versus 9% for Technical and Science Schools respectively. There was a smaller proportion of students who disagreed in the Science classes (Table 4.1).

Moreover, 8% of students felt that programming itself was a hinder to their understanding (Table 4.2). Some students perceived programming as difficult and may even have had an innate fear or loathing of the subject, which one teacher pointed out is difficult to treat during a single class:

> *You can't really reach students who have a deep-founded fear of programming [in a 70 minutes lesson, who think] "this is hard, I can't understand any of it" even though they may very well understand or there is no need for them to understand and they only have to think about the maths but their thoughts freeze on "I don't understand this".*

For Lesson 2 another category of no-answers was identified as "*pace too high*". 4% of answers in this lesson felt that the high pace was a hindrance to increasing their understanding of solids of revolution (Table 4.2). However, several interviewed students independently described the tasks as relatively easy even though they felt rushed.

52% of students in Lesson 2 named visualisation as the reason for increasing of their understanding (Table 4.2). This is one of the main differences in relation to Lesson 1, where visualisation was not even identified as a category. The former lesson had a much larger focus on constructing visual 3D-graphs that aid in understanding aspects of solids of revolution while the latter had a broader approach, meaning that this discrepancy was not unexpected.

Of the students that motivated their answer to the statement of having learned something during Lesson 1, 56% were categorised as *Deeper understanding*. None of the students had proved or seen proof of the fundamental theorem of calculus in class previously although some students had worked with the proof in their spare time. The categorisation of *Deeper understanding* mostly contains answers regarding the proof or its meaning, namely the connection between integrals and derivatives. Mathematical proofs are not a main part of mathematics until later courses in high school, and therefore most students found this unfamiliar.

19% of student answers to why the lesson increased their understanding of integrals in Lesson 1 were categorised as *Programming*, meaning that they thought that some direct aspect of programming increased their understanding of integrals (Table 4.2). Some of these answers named programming in mathematics and its' uses as a fresh perspective and something that they had not often or at all encountered before while some students viewed the introduction of programming into mathematics as a different means of teaching. One student said:

> *Not everyone is good at thinking mathematically but if you try another way of thinking more like that of a computer, step by step, then maybe it is a bit weird for some people but maybe it is great for others. Education should not be designed for one type of person.*

Other students also concluded that having different ways of teaching something is good since different people learn best in different ways.

For both lessons, only 6% of students answered that the *Lecture* or some part of the given explanations were what increased their understanding of the mathematics (Table 4.2). This categorisation is the only one which clearly doesn't involve any aspect of programming.

4.3.2 Takeaway from the lesson

When answering "*What is your takeaway from today's lesson*", most students (58% of answers) said that they have learnt something related to integrals or mathematics, see Table 4.2. There was a difference between lessons - for Lesson 1, 68% of answers

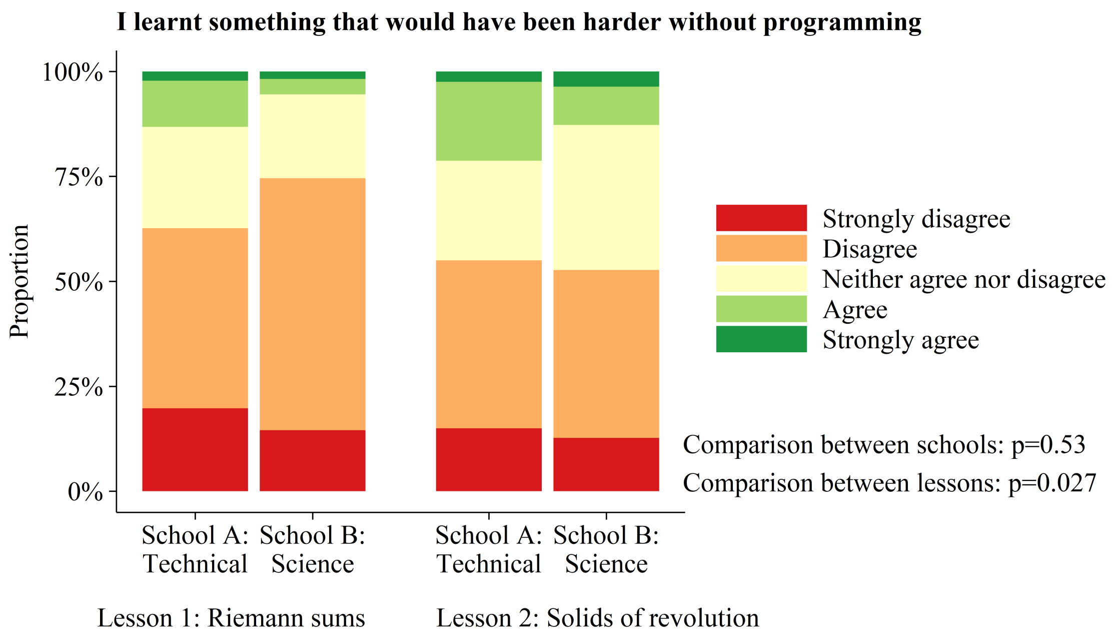


Figure 4.3: Student perceptions of the uniqueness of the contributions that programming had on mathematics understanding. The data in this figure is also found in Table 4.1 but with some related categories merged. All tests with $p < 0.05$ are considered statistically significant, where the p-value is the probability to obtain a result that is as least as extreme as the observed outcome given that no difference exists.

named integrals while for Lesson 2, 49% named mathematics. On the other hand, 37% of answers (32% for Lesson 1 and 42% for Lesson 2, see Table 4.2) point out that they learnt programming or code with them. Some students in Lesson 1 (10% of answers) emphasised that they had not learnt anything in the lesson. In contrast, some students in Lesson 2 (20% of answers) said that their main takeaway was the visualisation.

4.3.3 Programming as a Facilitator of Knowledge

There were 17% of Technology students versus 9% of Science students overall who thought that they had learned something that would have been more difficult without programming, see Table 4.1. Statistical significance ($p = 0.027$) points out that students gave higher grades on this question in Lesson 2 than in Lesson 1. The proportion of students who had not *learned anything that would be difficult without programming* (disagree) was smaller in the second lesson than in the first (see Table 4.1). In addition, qualitative analysis of free-text answers shows that the students have learned visualisation, numerical calculations, concrete representation and experimentation, see Table 4.2.

In this context, it should be noted that most students did not answer the question that was asked ("*What did you learn that would have been harder without programming?*")

but instead explained how programming helped them to understand or gain additional understanding of the concept of integral/solid of revolution. Most students who answered this question (75% of answers, see Table 4.2), thought that the visualisation helped them understand better. There were more students in Lesson 2 (solids of revolution) who thought that visualisation helped them. Comparing the lessons, the percentages were 83% for Lesson 2 and 64% for Lesson 1, see Table 4.2.

19% of answers to ("*What did you learn that would have been harder without programming?*") were defined as experimenting. Those students thought that experimenting during the lessons created more understanding. Programming gives students the opportunity to more easily change different values, functions, ranges and see how that affects the results. In addition, interviewed students said that programming facilitated work in mathematical subjects, see Figure 4.5. For example, two students stated that:

> *Programming works as a tool in mathematics for problem solving, visualising things and understanding why things behave the way they do.*
>
> *You can solve challenging mathematical problems with algorithms that you can code.*

Some students (30% of all answers) answered that programming can be an effective tool when applied to executing numerical calculations, see (Table 4.2). These students have mentioned one of the traditionally most important uses of programming in mathematical subjects.

Finally, some students (23% of answers in Lesson 1) thought that programming gave a more concrete representation of the integral concepts. This perspective was not found in Lesson 2.

Out of all students, 61% disagreed with the statement *I learnt something that would have been harder without programming* (Table 4.1). The reasoning of one student who thought this way is as follows:

> *The thing is that we had already understood how to calculate the integral with the help of rectangles. The only thing programming did was to visualise it even more but we had already seen a visualisation - so if I'm being completely honest - I don't quite see what programming did except maybe make it easier for us to experiment and visualise. But I imagine that most people - or at least my own opinion is that I already understood that before the lesson. So for me the programming was not very helpful in understanding the maths problem, it didn't contribute a lot.*

Table 4.1: Descriptive statistics of survey results and comparison between groups.

	Lesson 1: Riemann sums		**Lesson 2: Solids of revolution**			
	School A, Technical	**School B, Science**	**School A, Technical**	**School B, Science**	**School A *vs.* B**	**Lesson 1 *vs.* 2**
Total number of students	100	59	100	70		
Number of participants	91 (91.0%)	55 (93.2%)	80 (80.0%)	55 (78.6%)		
Mathematics course	Ma3c	Ma3c	Ma3c	Ma4		
I do programming on my leisure time[*]						
Never/almost never (<once a month)	45 (49.5%)	51 (92.7%)	30 (37.5%)	47 (85.5%)		
Rarely	25 (27.5%)	2 (3.6%)	28 (35.0%)	5 (9.1%)	A>B	
Often (≥ once a week)	21 (23.1%)	2 (3.6%)	22 (27.5%)	3 (5.5%)	p<.0001	
I have previous experience of programming in school[*]						
Never	0 (0.0%)	12 (21.8%)	0 (0.0%)	23 (41.8%)		
Occasionally	2 (2.2%)	33 (60.0%)	3 (3.8%)	23 (41.8%)	A>B	
Course in programming	89 (97.8%)	10 (18.2%)	77 (96.2%)	9 (16.4%)	p<.0001	
I had enough prior knowledge in programming[†§]						
Disagree	17 (18.7%)	26 (47.3%)	8 (10.0%)	28 (50.9%)		
Neutral	17 (18.7%)	13 (23.6%)	15 (18.8%)	10 (18.2%)	A>B	2>1
Agree	57 (62.6%)	16 (29.1%)	57 (71.2%)	17 (30.9%)	p<.0001	p=0.56
The code was easy to use and understand[†§]						
Disagree	13 (14.3%)	12 (21.8%)	6 (7.5%)	14 (25.5%)		
Neutral	23 (25.3%)	21 (38.2%)	20 (25.0%)	16 (29.1%)	A>B	2>1
Agree	55 (60.4%)	22 (40.0%)	54 (67.5%)	25 (45.5%)	p=0.0002	p=0.14
The mathematical content was easy to follow[†§]						
Disagree	13 (14.3%)	3 (5.5%)	2 (2.5%)	6 (10.9%)		
Neutral	20 (22.0%)	11 (20.0%)	15 (18.8%)	8 (14.5%)	B>A	2>1
Agree	58 (63.7%)	41 (74.5%)	63 (78.8%)	41 (74.5%)	p=0.74	p=0.060
It was fun to use programming in class[†§]						
Disagree	18 (19.8%)	19 (34.5%)	16 (20.0%)	14 (25.5%)		
Neutral	25 (27.5%)	17 (30.9%)	29 (36.2%)	13 (23.6%)	A>B	2>1
Agree	48 (52.7%)	19 (34.5%)	35 (43.8%)	28 (50.9%)	p=0.083	p=0.60
The lesson increased my understanding of integrals[†§]						
Disagree	21 (23.1%)	11 (20.0%)	23 (28.8%)	5 (9.1%)		
Neutral	26 (28.6%)	23 (41.8%)	22 (27.5%)	27 (49.1%)	A>B	1>2
Agree	44 (48.4%)	21 (38.2%)	35 (43.8%)	23 (41.8%)	p=0.58	p=0.95
I learnt something today that would be difficult without programming[†§]						
Disagree	57 (62.6%)	41 (74.5%)	44 (55.0%)	29 (52.7%)		
Neutral	22 (24.2%)	11 (20.0%)	19 (23.8%)	19 (34.5%)	A>B	2>1
Agree	12 (13.2%)	3 (5.5%)	17 (21.2%)	7 (12.7%)	p=0.53	p=0.027

Disagree = Strongly disagree or disagree, Neutral = Neither agree nor disagree, Agree = Agree or strongly agree.

All tests with $p < 0.05$ are considered statistically significant, where the p-value is the probability to obtain a result that is as least as extreme as the observed outcome given that no difference exists.

[*]Comparison between schools were performed using Wilcoxon rank sum test.

[†]Comparisons between schools were performed using val Elteren test, stratified by programming lesson.

[§]Comparisons between lessons were performed using val Elteren test, stratified by school.

4.4 Difficulties of Programming in Mathematics

During interviews with students and teachers some potential difficulties with using programming in mathematics were uncovered. These are presented in Figure 4.4. The named potential difficulties were categorised into four main groups: *cognitive load*, *varying prior knowledge*, *unclear purpose* and *additional difficulties.* In addition to the interviews, students also answered what part of the lesson they found the most difficult in the survey.

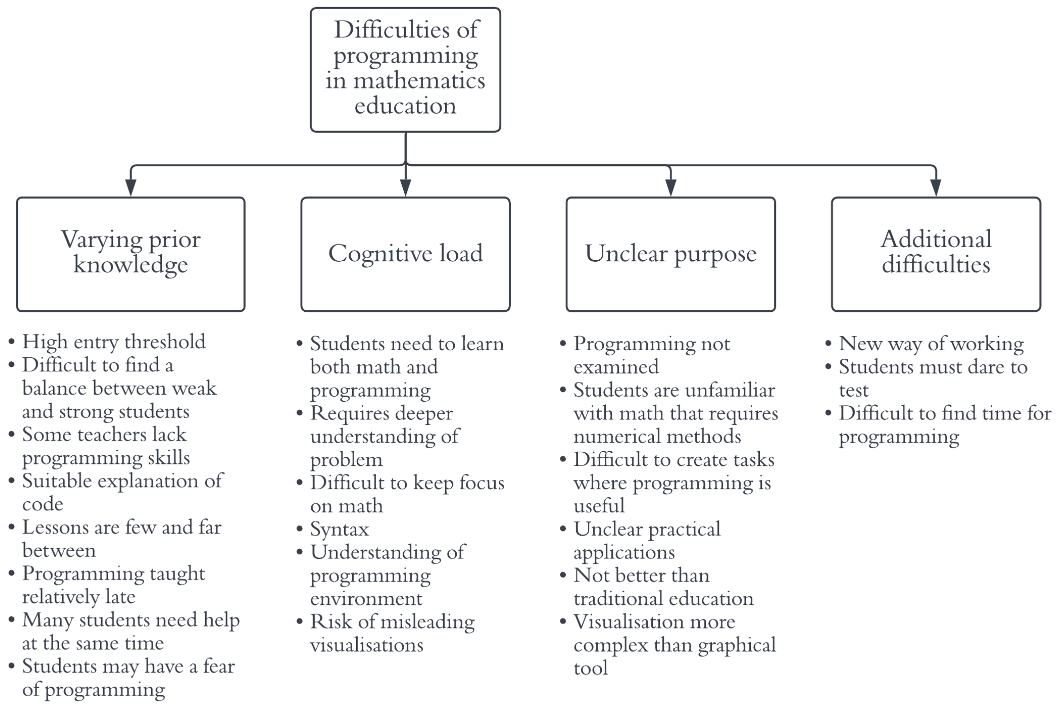

Difficulties of programming in mathematics education
Varying prior knowledge
Cognitive load
Unclear purpose
Additional difficulties
• High entry threshold
• Difficult to find a balance between weak and strong students
• Some teachers lack programming skills
• Suitable explanation of code
• Lessons are few and far between
• Programming taught relatively late
• Many students need help at the same time
• Students may have a fear of programming
• Students need to learn both math and programming
• Requires deeper understanding of problem
• Difficult to keep focus on math
• Syntax
• Understanding of programming environment
• Risk of misleading visualisations
• Programming not examined
• Students are unfamiliar with math that requires numerical methods
• Difficult to create tasks where programming is useful
• Unclear practical applications
• Not better than traditional education
• Visualisation more complex than graphical tool
• New way of working
• Students must dare to test
• Difficult to find time for programming

Figure 4.4: Potential challenges of working with programming in mathematics education, identified by analysis of interviews with students and teachers.

4.4.1 Most Difficult Part of the Lesson

The most frequently named difficult part of the lesson was the programming and code with 63% and 31% of student answers in Lesson 1 and 2 respectively, see (Table 4.2). Most of the students had only used programming occasionally during previous courses in mathematics, though there was a large variation between overall prior programming experience between students. An additional difference between the lessons was that 14% of students in Lesson 2 explicitly named the syntax as problematic while students in Lesson 1 found other parts of the programming difficult.

Another difficulty for students was the mathematics involved in the lessons. The percentage of students who thought that mathematics was the hardest part of the lesson was relatively similar for both lessons, with 20% and 15% for Lesson 1 and 2 respectively (Table 4.2).

Table 4.2: Categorisation of free text answers from surveys. Note that some answers contained elements of multiple categories, meaning that the percentages do not add up to 100%.

(a) Categorisation of free text answers regarding the survey that students took after Lesson 1.

The lesson increased my understanding of integrals	N	(%)
No: Already knew	38	(33.6%)
No: Easier without programming	8	(7.1%)
Yes: Deeper understanding	63	(55.8%)
Yes: Programming	21	(18.6%)
Yes: Lecture	5	(4.4%)
Total respondents	113	
What did you learn that would have been harder without programming?		
Visualisation	25	(64.1%)
Numerical calculations	16	(41.0%)
More concrete representation	9	(23.1%)
Experimentation	8	(20.5%)
Total respondents	39	
What did you find most difficult about today's lesson?		
Programming/code	64	(63.4%)
Mathematics	20	(19.8%)
Instructions	20	(19.8%)
Lack of time	12	(11.9%)
Total respondents	101	
What is your takeaway from today's lesson?		
Integrals	75	(67.6%)
Programming/code	35	(31.5%)
Nothing	11	(9.9%)
Total respondents	111	
What was good about the lesson plan?		
Structure	55	(50.0%)
Variation	33	(30.0%)
Interactive	29	(26.4%)
Progression	16	(14.5%)
Experimentation	10	(9.1%)
Tempo	7	(6.4%)
Total respondents	110	
What can be improved regarding the lesson plan?		
More time	31	(38.8%)
Clearer instructions	20	(25.0%)
Execution	18	(22.5%)
More challenging	11	(13.8%)
Less repetition	8	(10.0%)
Total respondents	80	

(b) Categorisation of free text answers from the survey regarding Lesson 2.

The lesson increased my understanding of solids of revolution	N	(%)
No: Already knew	41	(32.8%)
No: Easier without programming	11	(8.8%)
No: Pace too high	5	(4.0%)
Yes: Visualisation	65	(52.0%)
Yes: New representation	24	(19.2%)
Yes: Lecture	9	(7.2%)
Total respondents	125	
What did you learn that would have been harder without programming?		
Visualisation	47	(82.5%)
Numerical calculations	13	(22.8%)
Experimentation	10	(17.5%)
Total respondents	57	
What did you find most difficult about today's lesson?		
Programming	39	(31.0%)
Mathematics	19	(15.1%)
Lesson plan	19	(15.1%)
Programming syntax	17	(13.5%)
High pace	13	(10.3%)
Total respondents	126	
What is your takeaway from today's lesson?		
Mathematics	57	(48.7%)
Programming/code	49	(41.9%)
Visualisation	23	(19.7%)
Total respondents	117	
What was good about the lesson plan?		
Structure	69	(62.2%)
Execution	28	(25.2%)
Interactive	19	(17.1%)
Experimentation	16	(14.4%)
Progression	12	(10.8%)
Total respondents	111	
What can be improved regarding the lesson plan?		
More time	44	(49.4%)
Clearer instructions	17	(19.1%)
More challenging maths	16	(18.0%)
Code/programming	13	(14.6%)
Variation	12	(13.5%)
Execution	8	(9.0%)
Lesson goal	4	(4.5%)
Total respondents	89	

Additional named difficulties concerned the planning and execution of the lessons instead of content knowledge. These were made up of the categories *Lack of time* (12% of answers in Lesson 1) and *High pace* (10% of answers in Lesson 2) along with unclear *Instructions* (20% in Lesson 1) and *Lesson plan* (15% in Lesson 2), see Table 4.2.

4.4.2 Cognitive Load

Cognitive load refers to the increased mental strain of incorporating more components into the lesson. Students had to focus on programming in addition to the mathematics which the lesson was primarily about. One quotation that was representative for students and teachers regarding the cognitive load was the following:

> *I think that the [mathematical] concept is a bit difficult to understand the first time and programming is also a bit difficult. So combining two difficult things makes it extra difficult. However, I do think it is nourishing for our development.*

4.4.3 Varying Prior Knowledge

Varying prior knowledge refers to the problem of extreme variety in knowledge between students. For example, in one class experiences ranged from students only having experience of programming from the first programming course that they have studied to one student who started programming when he/she was 6 years old and who was employed part-time as a programmer in high school. This can cause a range of problems, one of which was illustrated by the following quote from a student:

> *Many others didn't quite understand the instructions. Several times you asked "do you understand?" and I was the only one who answered "yes I understand" but then my friends asked "what are we supposed to do?"*

4.4.4 Unclear Purpose

Another challenge when introducing programming in mathematics education was the *unclear purpose* of programming perceived among both students and teachers, see Figure 4.4. Many students and teachers found the role of programming in mathematics education unclear. Teachers generally did not test student knowledge of programming. One reason for this was that questions about programming have not been included on the National mathematics tests (*Nationella proven*). One teacher said:

> *When the National tests start evaluating programming, then we might see a change in the mathematics courses.*

4.4.5 Additional Difficulties

The usage of programming in mathematics education has brought with it some varying difficulties which do not belong to any previously defined categories. One such difficulty was the different way of working that programming requires compared to other mathematics teaching - there are often many different ways of arriving at the correct answer and students must improve their code in iterations and be willing to try and fail. Additionally, teachers found it difficult to make time for programming lessons and fit them into the curriculum. The different way of working is highlighted by one teacher with the following quote:

> *Programming is much more open in that way, sure you're supposed to arrive at an answer but the path there can look completely different and you have to understand what you are doing in another way.*

4.5 Possibilities of Programming in Mathematics

From a thematic analysis of interview summaries, five reasons as two why students and teachers were positive to programming in mathematics education were identified: *Automation of a process; Tool for problem solving; Focus on mathematics instead of calculations; Experiment and visualise;* and *New perspective* (Figure 4.5). These can be further grouped into the possibilities of programming to *Deepen understanding* and to *Facilitate work.*

4.5.1 Deeper Understanding

Programming can deepen student understanding by broadening the possibility to experiment and visualise, but also by providing a new perspective on the mathematics that they are working on. With a finished program, it is easy to change certain parts of the code and explore how that affects the result. This also holds true for visualisations created with programming. One student said the following:

> *It may be easy to calculate a volume [of a solid of revolution], but maybe you don't really understand why it is the way it is. Actually seeing a graph, moving it around and testing, that is better.*

Student understanding of mathematics may also increase by giving students an alternative perspective on the mathematics in question. It could be components such as algorithmic thinking, seeing relations to other subjects or the different way of seeing mathematical concepts that naturally accompanies the translation of mathematics to code. A student quoted that:

> *I would say that programming is useful to gain a new perspective on things, to increase understanding.*

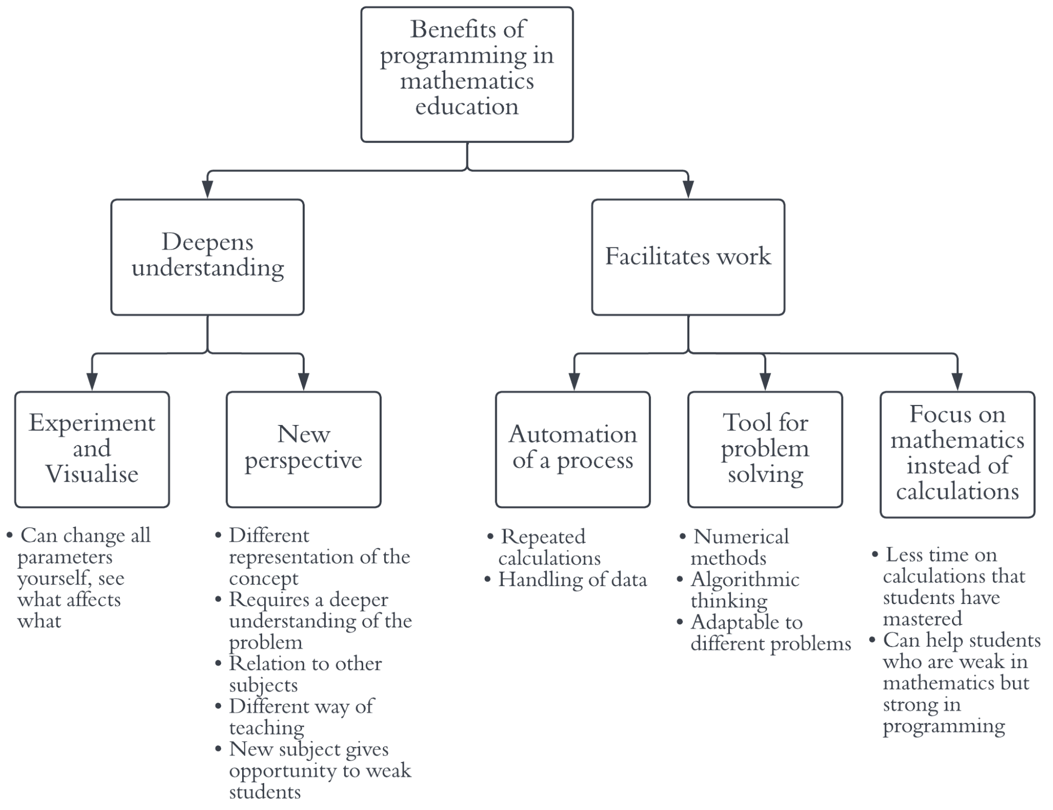

Benefits of programming in mathematics education
Deepens understanding
Facilitates work
Experiment and Visualise
New perspective
Automation of a process
Tool for problem solving
Focus on mathematics instead of calculations
• Can change all parameters yourself, see what affects what
• Different representation of the concept
• Requires a deeper understanding of the problem
• Relation to other subjects
• Different way of teaching
• New subject gives opportunity to weak students
• Repeated calculations
• Handling of data
• Numerical methods
• Algorithmic thinking
• Adaptable to different problems
• Less time on calculations that students have mastered
• Can help students who are weak in mathematics but strong in programming

Figure 4.5: Possible benefits of working with programming in mathematics education, identified by analysis of interviews with students and teachers.

4.5.2 Facilitation of Mathematical Work

Programming can also facilitate mathematical work by serving as a *Tool for problem solving*, *Automating processes* or by shifting *Focus to the mathematics instead of calculations*. Programming is adaptable to different problems and some can only be solved with numerical methods, making them very time-consuming to solve without programming. A teacher expressed the following:

> *[Programming] works great for problem solving, you can solve problems in different ways by changing a small part of the code. That increases understanding and students have the opportunity to use while- and for-loops which is impossible when working with pen and paper.*

Programming can also help students by not spending unnecessary time on calculations. One student phrased it as:

> *Most of the time, programming helps you skip the long, boring steps that you already know since five years back and then it becomes more about learning new things and not repeating old knowledge over and over again.*

5 Discussion

In this chapter, results from the study are discussed in relation to the three central research questions presented in Section 1.3. The chapter is thereby divided into three sections regarding *Previous Programming Experience* (Section 5.1), *Difficulties in Using Programming* (Section 5.2) and finally *Possibilities in Using Programming* (Section 5.3).

5.1 Previous Programming Experience

Results revealed that Technology students had much more experience in programming, both from school and their own leisure time. Furthermore, one of the main factors for a positive outcome of the lesson was that students found the code easy to use and understand. Previous experience both in school and during leisure time correlated with finding programming engaging and easy to use in the lesson, but was not found to be a factor for increasing mathematical understanding.

More Technology students than Science students agreed with the statement *the code was easy to use and understand* (64% vs 43%). The big difference between both disciplines can be related to that the Technology students had studied at least one programming course in the school where they may have been working with similar codes. However, most Science students were unaccustomed to working with programming.

A portion of Technology students who studied a programming course did not consider themselves to have enough prior programming experience to understand the lesson. In contrast, some Science students who never studied a programming course considered themselves to have enough experience. This may be because Science students had more knowledge about the mathematics involved. They were completely done with the mathematics taught using programming while Technology students were in the middle of working with this subject. Moreover, it may be that some Technology students were unaccustomed to how programming was used during the lesson because they had studied different programming languages. They may have been unfamiliar with the Python syntax. Furthermore, differences between different lessons could have some effects. An explanation as to why Science students who had not attended a course in programming considered themselves to have enough experience is that lessons were designed to teach mathematics and not programming. Therefore, students did not have to be good programmers, only to understand the code and manipulate it. In any case, some programming knowledge was needed to keep up and understand the lesson.

This result indicates that previous programming experience plays a role in mathematics education when programming is involved, but not a decisive role in the lessons that were designed for this project.

5.2 Difficulties in Using Programming

The main difficulties of using programming in mathematics education that were observed in this study were *Cognitive load, Varying prior programming knowledge* between students and teachers, *Unclear purpose* of why programming ought to be used in mathematics education, and a number of less specific *Additional difficulties*.

A small part of students believed that the use of programming was the reason that the lesson did not increase their understanding of integrals. One potential reason for this could be that many were uncomfortable in working with programming and that uncertainty accompanies any work involving programming. The introduction of a subject where they lack knowledge and experience into familiar mathematics creates uncertainty that affects their perceived understanding of integrals overall.

The lesson component which the largest share of students found most difficult was the programming and code involved. Significantly more students found the programming/code hard for Lesson 1 than Lesson 2. This is likely because of the different natures of the lessons - in Lesson 1, students were presented with the entire complexity of the code. On the contrary, Lesson 2 hid a lot of its complexity inside the help-functions that were created for students. This instead means that students found it harder to understand how the commands worked and how the syntax was formatted, but found the programming in itself easier.

The primary reason as to why students found programming to be hard seemed to be that they lacked experience. Especially the Science students were lacking experience since they only programmed during few and far between mathematics lessons, while Technology students may have been unused to the mathematical context and Python as a programming language.

Slightly more than half of students disagreed with the statement "*I learnt something that would have been harder without programming*". Bearing in mind that programming was used as a means to teach mathematics rather than in the traditionally supported ways of as a tool for solving problems or an end in itself, this is perhaps not very surprising. An important distinction is that this is not the same as students finding programming useless - these students simply felt that they did not learn anything new that "ordinary" teaching would be worse suited towards teaching.

Next, the four identified categories of difficulties *Cognitive load*, *Varying prior knowledge*, *Unclear purpose* and *Additional difficulties* are discussed in detail.

5.2.1 Cognitive Load

There are numerous reasons as to why the *cognitive load* increases when adding programming to the lesson. Students need to expend energy towards understanding code in addition to the effort already spent on trying to understand the mathematics. In contrast to the tools that students usually employ when working with mathematics i.e. pen and paper, programming requires less intuitive tools in the form of an integrated development environment (IDE) where they can execute code. Additionally, students need to have a basic understanding of the specific syntax that the programming language in question utilises, even if their only task is to manipulate premade code. Any element that students do not fully understand may cause uncertainty, frustration and stress. Programming can contain quirks that have to do with how a computer works that are quite complex to explain to new programmers, such as floating point rounding errors. Most of the time, it is desirable that students try to understand as much of the lesson material as possible since this creates a deeper understanding of the material, but it can become overwhelming and lessen focus on the mathematics.

In addition, solving a problem with programming requires a deep understanding of problem. Otherwise the student will not be able to construct a program to obtain the solution. At the same time, when you create a program for solving a problem, this increases your understanding of it and gives more familiarity of all its aspects.

Another perspective was expressed by some few students in Lesson 2 - that the pace of the lesson was too high. This lesson had an ambitious amount of material to cover which resulted in a relatively high pace and little time to discuss and reflect on questions. However, most interviewed students agree that the tasks were relatively easy. This should be a problem that can be fixed by adjusting the lesson plan as it is not necessarily tied to programming. Even so, this problem may have a relation to programming, since the amount of things that the student has to think about increases naturally when programming is integrated. Thus, the amount of material may increase, leading to an increased cognitive load.

A feature of using programming, or digital graphing tools in general is that graphs or visualisations are generally fast to create and open to experimentation. In some cases, students may choose functions or problems which do not match the situations that the teacher has planned to address and possibly fall outside the purpose of the lesson. This may confuse students that like to experiment, since the visualisations can become needlessly complex. When students were experimenting with different functions for solids of revolution, this became especially apparent. During the visualisation of the solid that is created between two rotating functions defined on an interval, students sometimes tested functions that intersected on the interval (see Figure 5.1). The figure thus created is not something that might be seen in

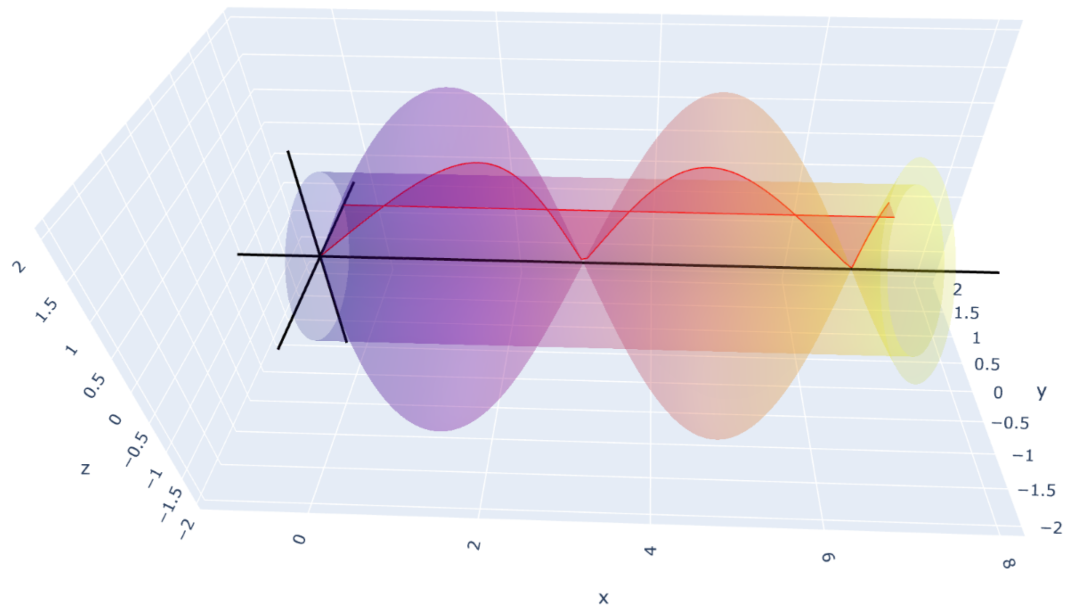


Figure 5.1: 3D illustration of two intersecting solids of revolution that students may encounter when experimenting with the code. Although mathematically feasible this kind of solid is not possible to observe in the physical world, which may cause confusion among students.

the physical world but more abstract and confusing because the surface plot of the rotating functions cross each other.

5.2.2 Varying Prior Knowledge

Another difficulty in utilising programming in mathematics is the large *variation in prior knowledge* between students or teachers, or both. Concerning students, programming has a steep initial learning curve which discourages students who are just starting to learn programming. Infrequent usage makes it difficult to overcome this threshold, which is unfortunate considering how infrequently programming is oftentimes used in mathematics education. Some of the reasons for this high entry threshold are: getting familiar with the integrated development environment (IDE), knowing the basics of the programming syntax, having to learn the functions and methods that programming offers, as well as having to think in a more algorithmic way.

Several students may have a fear of or aversion to programming, perhaps founded in their uncertainty in the subject or in stereotypes regarding programming or programmers. These negative attitudes may be difficult to overcome, especially during single programming lessons.

Some challenges regarding prior knowledge that affect students and teachers alike are: that lessons are few and far between; that some teachers lack programming skills and experience; that programming is taught relatively late in their education; difficulties in finding a balance between strong and weak students; and that many students may need help at the same time.

Infrequent and far between lessons do not allow teachers to assume that students have retained any programming experience since the last time they used programming. During every programming lesson in mathematics, the teacher has to spend time repeating basics that students likely would have learned if they used their programming knowledge more frequently than once in every mathematics course, which leaves less time for the material that is the actual focus for the lesson. This long time between lessons also means that teachers lack opportunities to practice and develop their programming skills and, like students, have to spend extra time refreshing their knowledge. This compounds the problem of some teachers lacking skills and experience in programming.

Aside from the apparent problem of not being able to teach as effectively that come with teachers lacking experience in programming, there are multiple potential problems that accompany this. Preparing for a lesson with programming will likely take much longer compared to preparing for a traditional lesson, swallowing up time from the teacher's already cramped schedule. Even if teachers have sufficient knowledge to teach programming, it is hard to create engaging lessons in programming where the utility of programming is obvious without extensive knowledge on the subject.

As previously mentioned, programming is a relatively new subject for many students since they lack previous experience and they start learning about it relatively late in their education. Programming is nowadays implemented as a part of mathematics in Sweden from early grades but since the change is relatively recent, current high school students have at most touched on programming in their later years of middle school. They belong to a generation caught in the middle of this sweeping change to the maths curriculum and therefore have less programming experience than future high school students.

During a lesson where programming is utilised in conjunction with mathematics, teachers have a challenging task to try to find a balance in the lesson such that experienced students are challenged while inexperienced students are not left behind. Since students have such differing amounts of prior knowledge - from essentially no experience to having significantly more experience than the teacher - this is very challenging.

Moreover, interviewed students and teachers pointed out that varying prior programming knowledge among students as one of main difficulties to using programming in mathematics education. Because of this, students in the classroom did the tasks at different paces. Students with a lot of previous programming experience felt that the tasks were easy and the lesson became boring, while those who lack experience felt that the tasks were very difficult and they needed more time to do small tasks.

Therefore, there are different perspectives, for instance some students say that the lessons were run at a low pace while others claim that the lessons were run at a very high pace. Programming experience plays a role in how much time it takes for students to complete the tasks.

5.2.3 Unclear Purpose

One of the most important results which was noted during interviews is the *unclear purpose* of programming perceived among both students and teachers. This difficulty is related both to lessons in this project and in general when programming is used in mathematics education. Many students and teachers wonder why programming is needed at all in mathematics education. This may be because most students and many teachers are unfamiliar with the kind of mathematics that require or is heavily simplified with programming.

This result coincides with previous research conducted by Jahnke (2020), where the purpose of programming was found to be unclear to teachers. She argues that the National Agency for Education must clarify the purpose by coming up with a clear curriculum, where students and teachers see how programming is used in mathematics and how it plays an important role to increase understanding mathematical subjects.

Another problem is that students' programming knowledge is not evaluated or tested. For students with a goal of simply passing the course or getting high grades, there is no motivation to learn any programming. Only students that either find it interesting or perceive a usage of programming in their future education or work are likely to expend energy trying to remember what is being taught.

5.2.4 Additional Difficulties

There were other difficulties that, while related to many of those previously discussed, did not fit under their overarching categories. These difficulties were: the difficulty for teachers to find time for programming, and the new way of working with mathematics were it is more important for students to dare to test their answers and work with trial and error.

One difficulty mentioned in the context of *varying prior knowledge* was that because many teachers lack experience of programming, it takes more time and effort for them to plan lessons that involve programming. Similarly, teachers explained that it was challenging to make time for programming in the already packed curriculum, as well as programming not being evaluated on the national tests. Thus, for teachers inexperienced with code, programming elements are arduous to plan and steal precious time from topics perceived as more important, leaving few reasons to integrate more programming elements than necessary.

Although programming has many similarities with mathematics, it is also very different in some aspects. Where mathematics is for students mostly about right and wrong, black and white, programming requires students to try solutions that they are uncertain about, oftentimes initially being wrong but improving their code in iterations. Many students dislike being incorrect, which can lead to them having an idea on how to solve a problem but not wanting to test it. This is likely most prevalent among students who perform well in other mathematics and often answer teacher questions correctly but are inexperienced with programming.

5.3 Possibilities in Using Programming

Only 44% of the students overall found that the lesson increased their understanding of the mathematical content involved. This result may be partly explained by the placement of the lesson in relation to other parts of the syllabus - the Science students in particular had already completed the topics on integrals of the course curriculum while the Technology students were relatively close to the end of the integral component, meaning that much of the material consisted of repetition.

A small number of students (6%) claimed that the *Lecture* was what increased their understanding. This categorisation is the only one which clearly doesn't involve any aspect of programming, meaning that a traditional lesson could do equally well to increase the understanding of integrals among students' in this category. In this context, it should be noted that lecture and lesson are different things. Lecture refers to the teachers' explanations, which is synonymous with *Explain* in the 5E-model.

There was a moderate correlation between students' perceived understanding and them finding the mathematical content easy to follow, or finding the code easy to use. This implies that to learn something from a lesson that combines both programming and mathematics, it is beneficial for students to find both the code easy and the mathematical content easy to follow. If students find the programming difficult, it stands to reason that it would be challenging for them to learn the mathematics content because of the focus that programming demands.

During the interviews, many students considered it good that the code and mathematics were mostly easy to understand. This is also the perhaps strongest argument against using programming to teach mathematical concepts - the more complex the concept becomes, the less students may understand when it is taught using programming as a method of teaching.

Most students' takeaway from the lesson were related to mathematics, while a smaller portion of students named programming. The fact that most students focused on the mathematics coincides well with the purpose of the lesson - using programming to learn about integrals and mathematics. It may be that the students who instead focused on programming thought that the purpose of the lesson was to become a better programmer, or to learn how programming can be used in mathematics, thus leading to their focus on programming.

5.3.1 New Perspective

As presented in Section 4.5 and Figure 4.5, some students appear to think of programming as an alternative representation or a different way of teaching. This stands in stark contrast to the nature of previous mathematics education, where students mostly solve problems with algebraic solutions and where numerical solutions are not needed. Despite this, many students seem to have developed a broader perspective of integrals and its applications during both lessons due to the programming involved. During Lesson 1, these advantages were made explicit when students first had to calculate the Riemann sum of six rectangles by hand. Later, they had to manipulate code that did the same thing automatically but could also increase the accuracy by increasing the number of rectangles for the given interval, which was much faster than attempting the same task by hand. Similarly, the advantages of programming were made clear during the second lesson when students had to visualise solids of revolution, which is very difficult to do without digital tools.

5.3.2 Visualisation and Experimentation

As mentioned in Section 4.5, students appear to have answered the question "*How did programming help you understand?*" instead of the actual question "*What did you learn that would have been harder without programming?*". In support of this claim are the identified categories for the answers to the question; *visualisation, numerical calculations, experimentation* and *concrete representation*. These are not things you learn during a single lesson but instead ways in which programming can support learning.

Programming helped some students understand the material through being a good tool for *visualisation* and *experimentation*. These two categories often go hand in hand with experimentation of code effecting a computer generated visualisation, though they can also individually strengthen learning.

A slightly larger share of students named visualisation as the reason for their increased understanding in Lesson 2 than in Lesson 1. This is likely because of the difference in nature between the lessons - in Lesson 2 the visualisation of solids of revolution was the focus for the lesson, while Lesson 1 utilised visualisations that students were more familiar with (such as graphs in 2D) and that are easier to generate. Another reason for this is that Lesson 2 focused on some of the first 3-dimensional graphs that students have worked with, further increasing the importance of the graphical representation. Through visualisation, students can see the matter from different points of view, something which is difficult to do with traditional tools in the form of pen and paper. Barring problems with the program, a digital visualisation is exact and easily experimented upon, providing students with many opportunities for variation in non-critical aspects in the vein of *Variation theory* (Ling Lo, 2012).

Thus far, visualisations generated through programming have only been discussed as opposed to hand-drawn graphs but not in relation to visualisation programs such as *GeoGebra* that also generate a digital visualisation. Programs such as these are often great for quickly generating graphs, although it is more difficult find out how to generate less common visualisations such as Riemann sums or solids of revolution. The ease of generating these visualisations is obviously a strength, but it can also be a detriment because students may not stop to think about what the visualisation represents. Programming, on the other hand, forces students slow down and understand *what* they are changing before they are able to change it. This can help them understand the problem on a deeper level while the increased complexity may also pose a barrier to weaker students' understanding.

5.3.3 Tool for Problem Solving

Many students and teachers considered programming to be a great tool for problem solving, with numerical methods being the main benefit of using programming as opposed to other methods of problem solving. Students rarely encountered problems that had to be solved numerically in their earlier education which may undercut the perceived importance of these methods. They may not have realised how many problems can be solved with numerical methods.

Programming is a very versatile tool with multiple possible paths for solving a single problem, which is nearly a requirement for tools to be useful for problem solving. For it to be useful, the student has to be somewhat familiar with programming. There is a level of freedom in choosing data structures and algorithms to complete the task at hand, but the largest amount of freedom may come from the translation of mathematics to code. This may also be one of the capital difficulties for new programmers since it requires extensive knowledge about both programming and the mathematics involved. However, if students manage to do this successfully, they can gain a deeper understanding of the problem and its solution.

It should be noted that neither lessons consciously attempted to include problem solving, since the focus for both lessons was increasing student understanding of mathematical concepts using programming as a means. Utilising programming for problem solving is generally more common and also how the National Agency for Education in Sweden espouses using it (Skolverket, 2022). Using programming as a tool for problem solving is a possibility that primarily teachers but also students have expressed during interviews.

5.3.4 Automation and Focus on Mathematics

Aside from facilitating mathematical work through being a suitable tool for problem solving, programming can help students *Focus on mathematics* and *Automate processes.*

Since a coded program executes all calculations for the user, less time is spent by the student on tedious routine calculations that they have memorised and performed dozens of times before. Instead, the student can spend time and energy on more worthwhile elements that better improve learning. Accompanying this benefit is of course the initial difficulty of creating the program that executes these calculations. Even so, according to the results of this thesis, students increase their mathematical understanding by attempting to program the problem unless they get stuck on a problem having to do purely with syntax or the code.

Similarly, programming can help automate processes. Constructing a program to automatically do something with a calculation is exactly what makes loops in programming so useful. Moreover, repeated and tedious calculations with multiple steps become trivial when a program has been constructed to automatically handle these intermediate steps.

5.4 Limitations and Future Research

Technology students had already started with the integral concept and knew many of the components of the lesson, while the Science students had completed the subject of integrals and solids of revolution. This is one of the main limitations which affects the work in this project. One thing that would be interesting to test is how well programming works to teach completely unfamiliar mathematics to students. In this thesis, it was difficult to find such a situation.

Moreover, another limitation in this study is the focus on the results of teaching only two programming lessons in integral subjects. It would be highly valuable to see the effects of programming when using it to teach the entire chapter on integrals. Another limitation is the lack of control group. Results of teaching integrals in the traditional way should be compared with teaching integrals using programming.

Finally, future research should investigate the affects of programming when it is a part of the course - tested and not only worked with on special occasions. New teaching materials seem to include slightly more material on programming than older versions, but most mathematics teachers leave out those assignments. Future research should further explore why many teachers skip programming in mathematics courses.

6
Conclusion

In this project, we have designed and evaluated two lessons to investigate the potential and use of programming in high-school mathematics education. Eleven lessons were held and data gathered in the form of survey responses along with student- and teacher interviews.

In this study, we found that programming can be a valuable medium for mathematics education in many different ways. It can provide a deeper understanding of integrals through experimentation and visualisation, but also by providing new perspectives. Moreover, programming facilitates working with mathematics. It is a suitable tool for problem solving and an ideal way to realise long and complex calculations. Programming can automate an enormous amount of repeated mathematical operations. Lastly, some students found it engaging to utilise programming during the created lessons.

Only a small part of students felt that the lessons did not increase their understanding of integrals. However, most students also felt that traditional lessons could teach the material at least as well as programming could. Only a small proportion of students thought that they learned something that would have been more difficult without programming.

The challenges of introducing programming in mathematics education can be summarised as an increased cognitive load, an unclear purpose and varying prior knowledge of students and teachers. However, the investigated study population had a good grasp of the mathematics involved. Therefore, additional research is required on the effectiveness of programming as a means for teaching when students are unfamiliar with the mathematics content of the lesson.

Previous programming experience plays an important role in making students feel that coding is easy and programming is fun to use in mathematics education. If students find the code easy to understand, that helps to increase their understanding of the mathematical content. In addition, a general interest and prior knowledge in programming may show some positive affects on students regarding programming as having a unique benefit when used in mathematics education.

In contemporary mathematics education in Sweden, programming is more of an afterthought than a well-integrated component. This is problematic because many of the uncovered difficulties of using programming in mathematics stem from its infrequent usage. These difficulties can likely be overcome through motivating teachers to more frequently include programming in mathematics education. This may in turn be accomplished by evaluating students' programming knowledge on the National tests and clarifying its purpose in mathematics education. Teachers will also need more time or space in the curriculum in order to implement more programming in mathematics education.

Bibliography

Agresti, A. (2003). *Categorical data analysis*. John Wiley & Sons.

Artusi, R., Verderio, P., & Marubini, E. (2002). Bravais-pearson and spearman correlation coefficients: Meaning, test of hypothesis and confidence interval. *The International journal of biological markers*, *17*(2), 148–151.

Ball, D. L., Thames, M. H., & Phelps, G. (2008). Content knowledge for teaching: What makes it special? *Journal of Teacher Education*, *59*(5), 389–407.

Bandura, A. (1969). Social-learning theory of identificatory processes. *Handbook of socialization theory and research*, 213–262.

Bråting, K., Kilhamn, C., & Rolandsson, L. (2020). Integrating programming in swedish school mathematics: Description of a research project. *MADIF12: the twelfth research seminar of the Swedish Society for Research in Mathematics Education, 14-15 Jan 2020, Linnaeus University, Växjö, Sweden.*

Bybee, R. W., Taylor, J. A., Gardner, A., Van Scotter, P., Carlson Powell, J., Westbrook, A., & Landes, N. (2006). *The BSCS 5E Instructional Model: Origins and Effectiveness*. Colorado Springs, Colorado, USA, BSCS. http://fremonths.org/ourpages/auto/2006/9/7/1157653040572/bscs5efullreport2006.pdf (accessed: 20.5.2022)

Chen, X., & Liu, W. (2022). The value of python programming in general education and comprehensive quality improvement of medical students based on a retrospective cohort study. *Journal of Healthcare Engineering*, *2022*.

Corrégé, J.-B., & Michinov, N. (2021). Group size and peer learning: Peer discussions in different group size influence learning in a biology exercise performed on a tablet with stylus. *Frontiers in Education*, *6*, 733663.

Eckert, A., & Hjelte, A. (2021). Positioning of programming in mathematics classrooms–a literature review of evidence based didactical configurations. *Sustainable mathematics education in a digitalized world*, 193.

Esaiasson, P., Gilljam, M., Oscarsson, H., Towns, A. E., & Wängnerud, L. (2017). Metodpraktikan : Konsten att studera samhälle, individ och marknad. Wolters Kluwer.

European Union. (2006). L394 Recommendation of the European Parliament and of the Council of 18 December 2006 on key competences for lifelong learning. *Official Journal of the European Union*, *49*.

Felder, R. M., & Brent, R. (2016). *Teaching and learning stem: A practical guide*. John Wiley & Sons.

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)*, *17*(12), 18–32.

Gillies, R., & Ashman, A. (2003). *Cooperative learning*. Taylor & Francis.

Google Colab. (2022). *Welcome to Colab!* https://colab.research.google.com/?utm_source=scs-index (accessed: 19.5.2022)

Helenius, O., Misfeldt, M., & Rolandsson, L. (2018). Om programmering i matematikundervisningen. https://larportalen.skolverket.se/LarportalenAPI/api-v2/document/path/larportalen/material/inriktningar/1-matematik/Gymnasieskola/448_matematikundervisningmeddigitalaverktygII_GY/del_01/Material/Flik/Del_01_MomentA/Artiklar/MA2_Gy_01A_01_omprogrammering.docx. (accessed: 28.4.2022)

Hellmark Knutsson, H., & Nilsson, F. (2015). Uppdrag att föreslå nationella it-strategier för skolväsendet. *Regeringsbeslut 1*, *2*.

Hothorn, T., Winell, H., Hornik, K., van de Wiel, M. A., & Zeileis, A. (2021). *Package 'coin'*. https://cran.r-project.org/web/packages/coin/coin.pdf (accessed: 20.5.2022)

Jahnke, A. (2020). Programmering i skolan: Vad, hur, när och varför?: Slutrapport från fou-programmet programmering i ämnesundervisningen. IFOUS.

Kim, S. (2015). *Package 'ppcor'*. https://cran.r-project.org/web/packages/ppcor/ppcor.pdf (accessed: 20.5.2022)

Lärteam matematik, NTI Johanneberg. (2020). *Dokumentation av programmering i matematik NTI Johanneberg*. Available on request to: kristina.nilsson@ga.ntig.se.

Ling Lo, M. (2012). *Variation Theory and the Improvement of Teaching and Learning*. Ineko AB.

Merrill, M. D. (2002). First principles of instruction. *Educational technology research and development*, *50*(3), 43–59.

Meyers, C., & Jones, T. B. (1993). *Promoting Active Learning: Strategies for the College Classroom*. Jossey-Bass.

Namdar, B., & Kucuk, M. (2018). Preservice science teachers' practices of critiquing and revising 5E lesson plans. *Journal of Science Teacher Education*, *29*(6), 468–484.

Phillips, D., & Soltis, J. F. (2015). *Perspectives on learning*. Teachers College Press.

Plotly. (2022). *Python Figure Reference: Single-Page*. https://plotly.com/python/reference/ (accessed: 19.5.2022)

Regeringen. (2010). 5§ skollagen 2010:800.

Skolverket. (2022). *Gymnasieskolan: Ämne - Matematik*. https://www.skolverket.se/undervisning/gymnasieskolan/laroplan-program-och-amnen-i-gymnasieskolan/gymnasieprogrammen/amne?url=-996270488%5C%2Fsyllabuscw%5C%2Fjsp%5C%2Fsubject.htm%5C%3FsubjectCode%5C%3DMAT%5C%26tos%5C%3Dgy&sv.url=12.5dfee44715d35a5cdfa92a3#anchor2 (accessed: 13.5.2022)

Smith, M. K., Wood, W. B., Adams, W. K., Wieman, C., Knight, J. K., Guild, N., & Su, T. T. (2009). Why peer discussion improves student performance on in-class concept questions. *Science*, *323*(5910), 122–124.

Stigberg, H., & Stigberg, S. (2020). Teaching programming and mathematics in practice: A case study from a swedish primary school. *Policy Futures in Education*, *18*(4), 483–496.

Tanner, K. D. (2010). Order matters: using the 5E model to align teaching with how people learn. *CBE—Life Sciences Education*, *9*(3), 159–164.

Tock, K. (2019). Google CoLaboratory as a platform for Python coding with students. *Robotic Telescopes, Student Research and Education Proceedings*, *2*(1).

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual.* CreateSpace.

Venebles, W. N., & Smith, D. M. (2022). *An Introduction to R.* https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf. (accessed: 20.5.2022)

Wiberg, E. (2020). Försöksverksamhet med samverkan kring praktiknära forskning (ULF), delredovisning 2020. https://www.ulfavtal.se/digitalAssets/709/c_709236-l_3-k_delredovisning-ulf-2020-gu.pdf. (accessed: 9.5.2022).

Wild, C. (1997). *The Wilcoxon Rank-Sum Test.* https://www.stat.auckland.ac.nz/~wild/ChanceEnc/Ch10.wilcoxon.pdf (accessed: 20.5.2022)

Wing, J. M. (2014). Computational thinking benefits society. *40th anniversary blog of social issues in computing*, *2014*, 26.

A

Student Survey

* Required

General questions and prior knowledge

1. Which high school are you studying at? *

◯ Technical high school

◯ Science high school

2. What grade are you studying in? *

3. Programming in your leisure time.

To what extent do you agree with the following statement where 1 = never or almost never (less than once a month), 2 = rarely (once a month but less than once a week), 3 = often (once a week or more). *

	1	2	3
I usually program in my spare time	◯	◯	◯

4. Programming in school

To what extent do you agree with the following statement where 1 = never, 2 = on occasion, 3 = I go / have taken a course in programming. *

	1	2	3
I have programmed during my education	◯	◯	◯

5. Prior knowledge in in programming before the lesson

To what extent do you agree with the following statement, where 1 = Strongly disagree, 2 = Disagree, 3 = Neutral, 4 = Disagree and 5 = Strongly disagree.
*

	1	2	3	4	5
I had sufficient prior knowledge in programming for today's lesson	○	○	○	○	○

Impressions from the lesson

6. Understanding of mathematics and programming during the lesson

To what extent do you agree with the following statement, where 1 = Strongly disagree, 2 = Disagree, 3 = Neutral, 4 = Agree and 5 = Strongly agree.
*

	1	2	3	4	5
The math during the lesson was easy to understand	○	○	○	○	○
The code was easy to understand and work with	○	○	○	○	○
It was fun to use programming in the lesson	○	○	○	○	○

7. The effect of the lesson on understanding integrals

To what extent do you agree with the following statement, where 1 = Strongly disagree, 2 = Disagree, 3 = Neutral, 4 = Agree and 5 = Strongly agree.
*

	1	2	3	4	5
The lesson increased my understanding of the integral concept	○	○	○	○	○

8. Motivate shortly your answer to question 7.

9. The contribution of programming

To what extent do you agree with the following statement, where 1 = Strongly disagree, 2 = Disagree, 3 = Neutral, 4 = Agree and 5 = Strongly agree.
*

	1	2	3	4	5
I have learned something about mathematics that would have been more difficult to understand without programming	○	○	○	○	○

10. If you answered 3-5 to question 9: What did you learn that would have been more difficult without programming?

11. What did you find most difficult about today's lesson?

12. What do you take with you from today's lesson?

13. What was good about the structure of the lesson?

14. What can be improved regarding the lesson plan?

15. What do you think about the lesson realisation?

16. I can participate in a short interview (about 15 minutes) that takes place at my school.

The interview takes place at your school and is adapted to your schedule. It is about what you think of the lesson. The goal is to capture all opinions and experiences from the lesson. Therefore, we want to be able to interview students with a great variety in programming experience.
*

◯ Yes, I would like to participate in interview.

◯ No, I do not want to be interviewed.

17. What's your name? Answer with first and last name. *

B

Interview Questions for Students

I. How have you programmed before?

- How much have you programmed before?
  - How much have you programmed on your leisure time versus in school?
  - How long have you been programming?
  - When did you start programming?
  - Have you used Python or a similar programming language before?
- What have you used programming for?
- Have you used programming in mathematics before? How?
- What are the similarities and differences between how programming was used during the lesson compared to how you are used to using it?

II. What do you think of today's lesson?

- How difficult was the mathematical content?
- How much of the mathematical content did you recognise before? What did you study through the lessons?
- How difficult or easy was the code to understand? Did you feel that you had sufficient programming skills before the lesson?
- Were the programming tasks reasonably challenging?
- Was the information clear?

III. Did the lesson increase your understanding of the integral concept /

solids of revolution?

- How did the programming contribute to your understanding of the integral concept / solids of revolution?

IV. Do you think that programming and mathematics work well together? How?

- How do you think that programming should be used in mathematics teaching? (Can you mention different ways: for teaching about concepts, as a tool for problem solving, for modeling the course of events, for different calculations, for numerical analysis, etc.)

- How useful is programming compared to other digital tools?

C

Interview Questions for Teachers

I. What are your overall impression on the lesson?

- What do you think about the lesson plan?
- What do you think about the lesson realisation?
- Which things worked well and which worked less well?

II. How did you experience the classroom climate?

- How well do you think the students understood the lesson?
- How interested were the students?
- Did you experience any difference between different groups of students? If so, which groups and why?
- Did you experience any difference between students with different programming knowledge concerned understanding and interest in mathematics? Have you experienced this before when you work with programming?

III. Do you find that programming helps students understand mathematics better?

- How well do you think the lesson plan was suitable to teach the integral concept / volume of revolution?
- In what way did the programming affect the students' understanding of the integral concept / volume of revolution during today's lesson?
- What mathematics skills do you think programming can help develop?

IV. Do you often use programming in mathematics teaching at school? How often?

- What programming languages have you used in your teaching?
- In what way do you use programming in mathematics teaching?
  - Do you use programming as a tool for problem solving?
  - Do you use programming as a means of teaching concepts?
  - What differences can arise when programming is used as a means for teaching compared to as a tool for problem solving?

V. What is your previous programming experience?

- How long have you been programming?

VI. Do you find it difficult to introduce programming in math lessons?

VII. What difficulties are there when you as a teacher use programming in mathematics education?

VIII. What attitude do you feel the students have to use programming in mathematics education?