Digital Experiences in Mathematics Education
https://doi.org/10.1007/s40751-024-00141-0

Computational Thinking in Secondary Mathematics Education with *GeoGebra*: Insights from an Intervention in Calculus Lessons

Christos Chytas[1,2] · **Sylvia Patricia van Borkulo**[1] · **Paul Drijvers**[1] · **Erik Barendsen**[2,3] · **Jos L. J. Tolboom**[4]

Accepted: 15 March 2024
© The Author(s) 2024

Abstract

Nowadays, mathematics teachers in K–12 strive to promote their students' mathematical knowledge and computational thinking (CT) skills. There is an increasing need for effective CT-embedded mathematics learning material and a better understanding of students' views toward them. In this work, we present the results of a research study, which included the design of a six-lesson learning activity aimed at fostering 16- to 17-year-old secondary students' CT skills in calculus lessons using the dynamic mathematics software *GeoGebra*. Our goal was to investigate how students experienced the CT-embedded calculus lessons with *GeoGebra* and what challenges they faced during their interaction with the learning material and software. We collected and analyzed data from students' code in *GeoGebra*, workbooks, semi-structured interviews, and questionnaires. Our findings suggest that most students mastered using CT concepts in calculus activities to a satisfactory degree and could reason about their computational solutions using *GeoGebra* and the generated graphs. Students' understanding of the mathematical content knowledge introduced was essential to complete the lesson series successfully and unnoticed gaps in prior knowledge emerged. Our study shows that students appreciate the CT-embedded calculus lessons and *GeoGebra*'s exploratory approach to mathematics problems when provided with appropriate support. We conclude that an integrated approach to mathematics education and CT is viable and can contribute not only to fostering CT but also to increasing interest in mathematics.

Keywords Calculus education · Computational thinking · *GeoGebra* · Mathematics education · Mathematics software

Extended author information available on the last page of the article

Published online: 12 April 2024

Introduction

There is a widespread and growing agreement among academics and educators that computational thinking (CT) should be taught to everyone, and that it is an important part of scientific literacy. In her impactful article, Wing (2006) argues that CT should be fostered as a basic literacy skill like "reading, writing and arithmetic" (p. 33). The popularization of Wing's article led to global efforts to embed CT in school and out-of-school learning environments. Moreover, there are increasing calls to promote CT in computing and other STEAM (science, technology, engineering, the arts, and mathematics) subjects, including humanities and arts (Perković et al., 2010).

Since informatics/computer science courses are often elective across countries and states (Yadav et al., 2022), integrating CT into STEAM subjects can potentially broaden participation in computing and provide a more realistic representation of the domains of science and mathematics as potential future career options. For this reason, many researchers see CT as a means to increase participation in computer science and to integrate computing into different disciplines (Weintrop et al., 2016), which offers promising opportunities for promoting interdisciplinary learning. The integration of CT into mathematics education is a natural fit that enables real-world applications through computational tools and provides opportunities for developing problem-solving skills in a systematic way, cognitive processes, and transposition (Kallia et al., 2021). Additionally, in contrast to computing, mathematics has a long-standing tradition as a mandatory subject in formal education, and integrating CT into mathematics education ensures that students from historically underrepresented backgrounds in computing can gain more exposure to the necessary skills and mindsets essential to thrive in a digital and technology-driven world.

As powerful computational tools become commonplace in mathematics and science, there are promising opportunities for fostering CT in non-programming tasks and unplugged learning activities (Dagienė & Sentance, 2016). Modern mathematics tools intended for education offer opportunities for fostering CT within mathematics learning activities aligned with current curricula and educational policies. Additionally, they provide fruitful visualization and interactivity opportunities for learning (Adelabu et al., 2019). The use of educational technology to foster CT dates decades back to the vision of progressive educators for using powerful programming tools that are accessible to a wide variety of students (Papert, 1980). According to Wilensky et al. (2014), such technological innovations in education allow students to explore mathematical concepts and "create" mathematics themselves.

This work builds on existing efforts for integrating CT into calculus lessons. There is little research on how secondary students perceive such CT-embedded learning activities and how these can be integrated into mathematics education. Overall, academic literature on the integration of CT into mathematics education shows promising results in terms of both learning gains and students' motivation. However, even though studies report findings on CT integration, they seldom

use Wing's (2006) working definition and do not explicitly address specific CT aspects such as abstraction and decomposition (Lv et al., 2023) and how students used such skills. Additionally, Rich et al. (2020) highlight the necessity for teachers to ensure students are well-acquainted with classroom digital tools and to be equipped with diverse and appropriate didactic approaches for teaching both computational and mathematical aspects effectively.

For this reason, integrating CT aspects into calculus lessons is a challenging task that requires designing, testing, and refining the developed material before being implemented into curricula. This work presents the results from a research study on a CT-embedded calculus lesson series (van Borkulo et al., 2021). The lesson series took place in mandatory mathematics lessons for calculus in the Netherlands (referred to as Mathematics B in the Dutch educational system). Mathematics B deals with more theoretical aspects of mathematics, especially algebra, calculus, and geometry. The subject is particularly suitable for students considering studying in scientific fields, enabling them to apply for more STEM-oriented study programs in tertiary education. Our study followed a mixed-methods approach to answer the following research question:

> RQ: How do students experience a *GeoGebra*-based intervention aimed at fostering CT in calculus lessons and what challenges do they face when engaging in CT-embedded activities?

To tackle the research question at hand, we examine the feasibility of implementing CT-embedded calculus lessons in youth's formal education. Moreover, we contribute to existing efforts in developing computationally rich learning experiences that allow educators to bring CT into calculus lessons using accessible yet powerful tools like *GeoGebra*.

Theoretical Background

CT skills are now widely considered essential for everyone. Educators in K–12 strive to foster their students' CT skills and digital literacy to prepare them for professional life and active participation in society. According to Wing (2006), CT involves aspects fundamental to computer science, but it is not a skill explicitly targeted at technical experts:

> Computational thinking is a fundamental skill for everyone, not just for computer scientists [...] involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. (p. 33)

CT research is growing fast, but there is no consensus on what CT is. In later work, Wing (2011) revised the definition as follows:

> Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out. (p. 20)

Early research on CT focused on elements such as abstraction, decomposition, pattern recognition, algorithmic thinking, and generalization (Selby & Woollard, 2013). Besides the different definitions of CT in academic literature, it is widely accepted that CT involves processes fundamental to computer science and computational problem solving, such as making abstractions of the most critical elements of a problem (Wing, 2017), algorithmic thinking (Futschek, 2006) which focuses on understanding and designing algorithms to solve problems, decomposing problems into smaller, more manageable parts (Rich et al., 2019), and pattern recognition to identify common characteristics between problems (Wing, 2006). For more than a decade, CT research has been evolving, and researchers address additional CT aspects like automation (Lee et al., 2014) and modularization (Atmatzidou & Demetriadis, 2016).

Another way of CT classification is included in the CT assessment framework of Brennan and Resnick (2012). The authors classified CT aspects in terms of computational concepts (e.g., loops), computational practices (e.g., testing and debugging), and computational perspectives (e.g., how learners connect computational activities with their lives). These aspects are fundamental in computing and especially programming.

Weintrop et al. (2016) argued about the significance of CT in mathematics and science and developed a taxonomy of CT practices in these fields. These practices include data practices, modelling and simulation practices, computational problem-solving practices, and systems thinking practices. Additionally, there is consensus that CT is an integral part of the STEM disciplines (Henderson et al., 2007), and integrating computing and CT into such subjects in K–12 is a natural fit. Science and mathematics are becoming increasingly computational, and CT (Wing, 2017) is now considered a scientific practice in education (Weintrop et al., 2016). Fostering CT in K–12 is an efficient way to prepare future scientists and responsible citizens in an increasingly computational world. Integrating CT into science classes, e.g., by using tools for agent-based modelling in science education, could equip more students with CT skills, considering that computer science is still an elective course for many students in secondary education (Wilensky et al., 2014).

Progressive ideas for including computing in education have a long history, dating back to the 1960s and 1980s when Alan Perlis argued that it is essential to introduce students of all disciplines to the theory of computation (Guzdial, 2008). In addition, Papert (1980) envisioned children and youth developing procedural thinking skills and learning programming and mathematics with Logo. Integrating CT into STEM subjects is often linked to higher learning gains in the respective subject. For example, a study in CT-embedded mathematics lessons for sixth graders by Calao et al. (2015) showed that integrating CT into mathematics lessons can lead to statistically significant increases in the learning outcomes of students, in terms of mathematical processes such as modelling, problem formulation, and problem-solving and reasoning among other processes. In general, educational mathematics software is linked with higher learning gains in mathematics and fostering mathematical and computational thinking (van Borkulo et al., 2021).

A study on the effectiveness of the educational mathematics software *GeoGebra* showed that high school students achieved better learning outcomes and

welcomed its use, which broadened their perspectives about mathematics learning (Arbain & Shukor, 2015). According to the National Council of Teachers of Mathematics, technology plays a key role in modern mathematics education:

> An excellent mathematics program integrates the use of mathematical tools and technology as essential resources to help students learn and make sense of mathematical ideas, reason mathematically, and communicate their mathematical thinking. (Brahier et al., 2014, p. 656)

Digital tools are becoming commonplace in K–12 as vehicles to explore and verify mathematical ideas. Moreover, visualization and interactive elements can enhance learning and provide excellent opportunities to promote mathematical thinking (Drijvers, 2018) and CT (van Borkulo et al., 2021). Digital tools for CT-embedded mathematics learning activities in different grades include programming languages like *Python* (Jenkins et al., 2012), programming languages that are more common in education like *Scratch* (Calao et al., 2015), Logo-based tools like MaLT (Kynigos & Grizioti, 2018), and spreadsheets (Sanford & Naidu, 2016; van Borkulo et al., 2023), among other tools.

In line with the educational practice discussed above, our lesson series focuses on fostering pre-university students' CT skills using accessible computational tools that many mathematics teachers in the Netherlands are already familiar with. CT skills can be fostered with or without digital tools and can go far beyond computer programming (Bell & Lodi, 2019; Caeli & Yadav, 2020). To provide a comprehensive understanding of CT in the context of this study, we use the revised definition of CT by Wing (2011, p. 20): "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out." In formulating a problem, a person needs to understand and define the problem in a clear and precise manner, often by breaking it down into smaller sub-parts. Expressing such a problem in ways that a computer, human, or machine would understand requires appropriate use of language (e.g., appropriate syntax in the case of computer programming languages or a mathematical formula in a mathematics context) and the design of algorithms, as a set of instruction steps to solve the potential problem at hand.

We position CT as a cognitive framework that comprises key CT elements universally accepted across academic literature (Dong et al., 2019; Kynigos & Grizioti, 2018): decomposition, pattern recognition, abstraction, and algorithmic thinking that are likely to be facilitated through the basic logical structures of sequence, selection, and loops (van Borkulo et al., 2021) through computational concepts (e.g., iteration, conditional statements, and variables, among others). These provide opportunities to engage students in developing computational practices (practices they develop when working with computational concepts such as testing and debugging) and form computational perspectives (Brennan & Resnick, 2012). By focusing on these core aspects, we have operationalized our study to the learning goals of our developed activities which provided us with a concrete foundation to investigate the challenges students encounter and potential mathematics content knowledge gaps they might have. The chosen aspects are intended to provide a working definition of CT for our study and do not aim to

discount the importance and relevance of other CT dimensions across academic literature.

Our work will hopefully be of value to educators and mathematics teachers who wish to address CT aspects in calculus lessons.

Method

We designed and implemented CT-embedded calculus learning activities in four classes of 16- to -17-year-old secondary students following an educational design research approach (Gravemeijer & Cobb, 2006; Yazan, 2015) that focuses on designing and evaluating CT-embedded calculus activities. Below, we present the educational context and provide information about the study's participants. Then, we describe the learning activity, the materials/tools used by the student participants, and our design rationale. Finally, we present the data collection and analysis procedures.

Study Context

For three of the four classes, the intervention took place in a physical classroom setting at three schools in the Netherlands. The intervention took place online for the fourth class to prevent the spread of COVID-19. The intervention that took place online consisted of five 45-min calculus lessons (including post-experiment activities, e.g., interviews and filling in questionnaires), and the interventions that took place physically consisted of six lessons of 50 min. The lesson series took place in mandatory mathematics lessons for calculus (referred to as Mathematics B in the Dutch educational system), and the students could work alone or in pairs.

There are different mathematics courses in secondary education in the Netherlands, including mathematics A and B. Mathematics B (or Wiskunde B) is addressed to pre-university students in the VWO (Voorbereidend Wetenschappelijk Onderwijs) education and prepares them for studying in scientific fields and enabling them to apply for more STEM-oriented study programs in tertiary education. It includes topics like algebra, geometry, trigonometry, and calculus. The course also includes more advanced content like functions, derivatives, integrals, and differential equations.

Description of the Lesson Series

The lesson series was developed as part of a larger project on introducing computational thinking in mathematics education. The calculus content covered in the lesson series focused on the perpendicular bisector, focal points, tangents, and zeros of functions, among other topics. The students were expected to use CT concepts (e.g., variables, conditional statements, and iterations) and practices (e.g., debugging,

testing, and evaluating) to find general solutions to calculus problems using the *GeoGebra* environment.[1]

The research team supported the teachers during the co-design and implementation of the lesson series in the Mathematics B classroom. *GeoGebra* is a popular software program in Dutch education, and the teachers in our study were already familiar with it, having used it previously in their lessons. Teachers' prior experience with *GeoGebra*, and GeoGebra being accessible (free, open-source) and providing visualization and interactivity possibilities, were determinant factors in choosing this tool.

Digital Tools: *GeoGebra*

GeoGebra can be downloaded and used as software or directly used online on its official website (geogebra.org). It is an accessible and open-source mathematics tool, which makes it an attractive choice for teaching calculus, geometry, and algebra (among other subjects) in school and out-of-school settings in primary, secondary, and tertiary education. *GeoGebra* allows students to generate points, lines, segments, and vectors, among other mathematical representations. Such mathematical representations can be dynamically altered afterward. It enables the use of variables for storing mathematical objects (e.g., numbers, points, and line segments) that can also be modified via buttons and input fields. Furthermore, it is also possible to use *GeoGebra* to introduce computational concepts in computing education, e.g., by using iteration lists and conditional statements. *GeoGebra* is gaining increasing popularity among mathematics teachers in the Netherlands, which made it an appealing choice for our research. Its advantages are its interactive graph generation capabilities, which allow learners to explore mathematics concepts through visualization (Adelabu et al., 2019).

Learning Materials: Workbooks

The learning activity included using a workbook that we developed and addresses CT and mathematics content using *GeoGebra* as described in Table 1, which presents the mathematics content and CT focus of the learning activity with brief examples. The successful completion of the developed activities required both mathematical and computational thinking. Employing abstraction, decomposition, algorithmic thinking, and generalization skills was key to efficiently completing the designed tasks. The workbooks with the activities served as a scaffolding tool for tackling the CT-embedded calculus problems that the students were asked to solve. The workbooks also included a short introduction to the *GeoGebra* environment and hints for tackling common problems.

Chapters 1–4 focused on using conditional statements, while chapters 5–7 focused on combining conditional statements with iterations. To generate the

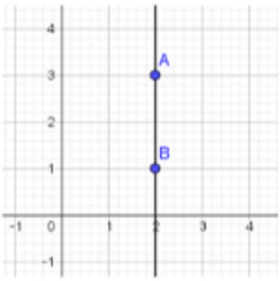
[1] The activities can be found in the following link: https://www.fisme.science.uu.nl/toepassingen/29236/.

Table 1 Mathematics content and computational focus in the respective chapters

Chapter	Mathematics content	Computational concepts focus	Example
Chapter 1	Line through two given points	Variables, conditional statements, and mathematics operations	Calculation of the slope of a line through points A and B
Chapter 2	Perpendicular bisector	Variables, conditional statements, and mathematics operations	Considering the case that the line might be vertical, i.e., points A and B are directly above each other
Chapter 3	Center of gravity	Variables, conditional statements, and mathematics operations	Determine the center of gravity of a general triangle equation considering the exception cases
Chapter 4	Tangent to a parabola	Variables, conditional statements, mathematics, and logic operators	Create a general parabola equation and test the solution by creating a tangent
Chapter 5	Bundles of tangents to a parabola	Variables, conditional statements, iterations, mathematics, and logic operators	Using iteration to generate a bundle of tangents to specific and general parabola equations
Chapter 6	Tangents to various graphs	Variables, conditional statements, iterations, mathematics, and logic operators	Computational experimentation for creating a bundle of tangents to different graphs (e.g., a root function and free choice graphs)
Chapter 7	Final task (not mandatory)—Newton–Raphson method	Variables, conditional statements, iterations, mathematics, and logic operators	Implementation of a root-finding algorithm which produces successively better approximations to the roots (or zeroes) of a real-valued function

Exceptions to the Algorithm

There is one more difficult case left: a vertical line. If *A* and *B* are directly above each other, it has no slope. Move point A directly above point B so they have the same x-coordinates. What do you notice in your equation?



In this case, you can use the 'if-then-else' statement from the world of programming.
This statement works as follows:
IF *(the x coordinates are equal)* THEN *(the line is vertical...)* ELSE *(the equation of the line is every other case i.e y=mx +c...).*

1.5
In your own words, see if you can write the if-then-else algorithm below.

IF .....

THEN .....

ELSE .....

1.6
Edit your GeoGebra commands, so that the line through *A* and *B* is also drawn in this exceptional case.
Use the IF statement in GeoGebra.
And test your solution. **(T)**

Fig. 1 Example of a workbook activity

intended graphs, the students needed to be precise and fully understand the concept of iterations and conditional statements, as well as the respective mathematics content.

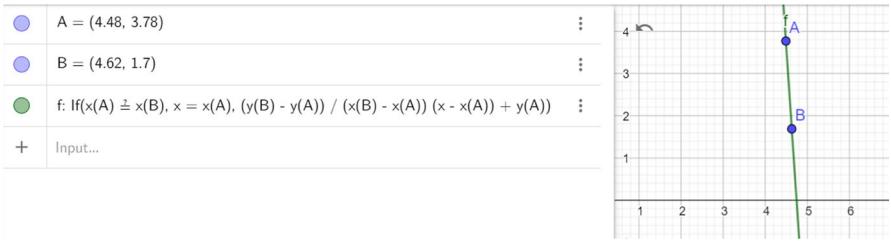


Fig. 2 *GeoGebra* generated interactive graph—a line through two given points

In Figs. 1 and 2 below, we provide examples of the workbook tasks and the generated graphs of students in the *GeoGebra* environment. In the task illustrated in Fig. 1, the students need to determine if the line is vertical and then graph the line accordingly. The students need to recognize that a vertical line occurs when two points have the same x coordinate and that when this happens, the line's slope is undefined and the line equation cannot be written in the form of $y = mx + b$. By using the conditional statement "if–then-else," they will check if the x coordinates of point A and point B are equal. If they are equal, they declare that the line is vertical. Otherwise, in the case that the x coordinates are not equal, they will use the slope-intercept form to define the line. The students are expected to transform their pseudocode in their workbooks into functional code in *GeoGebra* using the hints and syntax provided in the workbooks. After writing the code in the *GeoGebra* environment, they can execute it and test it. If necessary, they can make adjustments or debug their code until they get the desired outcome.

In chapter 6, the students could computationally experiment with their own functions and find general solutions to the calculus problems they chose. The seventh chapter about the Newton–Raphson method was designed as the most challenging task. It required using iterations/macros and a deep understanding of the aforementioned method to calculate the zeros of functions. The students were encouraged to write a report about this method where they explained how it works, but it was not mandatory.

Each chapter involved generalizing from a specific case to a more general one using parameters/variables and conditional statements. An example would be creating a general solution in which the equation changes when students drag points A and B. Therefore, students should consider the special cases and use appropriate conditional statements to generate the graphs and possibly evaluate them by dragging the points and testing their equations.

Design Rationale

A team of mathematics and computing education teachers and researchers co-designed these activities to address specific CT aspects, as illustrated in Table 1.

The design of the lesson series offers computationally rich and extensive opportunities for fostering key CT elements as it encompasses essential computational concepts such as iterations, conditional statements, and variables (see Table 1). These

Table 2 Computational thinking aspects in focus and intended student behavior

CT skills	Description of CT skill	Intended student behavior
Decomposition	Breaking down a problem into smaller parts	Calculating the sub-parts A and B in the standard form for linear equations in two variables: $Ax+By=C$
Pattern recognition	Pattern recognition involves observing and analyzing data and situations to identify patterns that can be critical in solving similar problems	Identifying graphical and coding patterns that can be reused (e.g., reusing parts of code from previous tasks)
Abstraction	Abstraction involves ignoring unimportant information to focus on what is important for solving a specific problem	Separate important from redundant information in specific activities (e.g., identify relevant variables, constants, and relationships between them)
Algorithmic thinking/design	Algorithmic thinking and algorithm design involve understanding and creating algorithms to solve a problem or complete a task in a way that others would achieve the same result if they followed the exact steps	Using the basic logic structures (sequences, selections, and loops), e.g., using an iteration list for generating 50 tangent lines instead of writing the same code 50 times)

concepts facilitate tasks like decomposition, pattern recognition, abstraction, and algorithmic thinking (see Table 2), which are universally accepted core CT elements across academic literature (Dong et al., 2019; Kynigos & Grizioti, 2018). Due to time and resource constraints, we did not directly observe every individual student's progress with the developed CT-embedded activities, but we employed a structured assessment approach for evaluating their learning outcomes on the design tasks and the specific CT elements at play. Tables 1 and 2 systematically describe which chapters corresponded to which computational concepts with examples (Table 1) and how students were expected to apply specific elements of CT that were needed in all the chapters successfully to complete the developed activities (Table 2). Table 2 also describes the CT skills required in the lesson series and the intended student behavior.

In the developed lesson series, the activities ask students to transfer their CT-embedded activities from their workbooks into *GeoGebra*. The workbooks address CT aspects of the activities as well, but transposing the activities to *GeoGebra* enables students to experiment and test their solutions in real-time and adjust them where needed. To do that, the students had to use the basic logical structures to translate the written mathematics exercises into a format understood by *GeoGebra*.

Most parts of decomposition are already offered as a scaffold so that the students can focus on understanding mathematics content and using the logical structures of sequence, selection, and loop to solve the computational problems of the lesson series. The students had to break equations apart by calculating the slope or other sub-parts of equations, decomposing iteration lists, formulating conditional statements, and adjusting variables/parameters to consider special cases when creating general solutions. The students were expected to use pattern recognition to identify, reuse, and generate the intended graphs in the *GeoGebra* environment. Students need to create functions and equations which represent mathematics concepts by abstracting the essence of the concepts at hand into digital representations (abstraction) that can be manipulated and reused.

The activities require students to use algorithmic thinking/design skills to translate the calculus activities of the workbooks into computational solutions in the *GeoGebra* environment. They need to use the basic logical structures (sequence, selection, and loop), which are integral elements to solving every algorithmic problem. The logical structure of sequence can be used to translate the steps of the workbooks into computational steps in *GeoGebra*. The logical structure of loop can be used to generate iterative graphs (chapters 5–7) like a tangent bundle in the respective activities. The selection structure can be used to consider the special cases of equations that can lead to general calculus solutions (e.g., considering if the fraction's denominator in an equation is zero). Considering such special cases requires crucial elements of algorithmic thinking and generalization. The workbooks also provide hints for considering the special cases in the first chapters.

The developed activities begin with simple tasks and progress to more complex ones, accompanied by hints and scaffolding so that the students can familiarize themselves with *GeoGebra* and proceed to more sophisticated activities. The activities were designed to enable students to complete independently the tasks at hand individually or in small groups, with minimal intervention from the teacher. The

teacher takes the role of the facilitator, supporting students as needed while refraining from directly disclosing answers and writing code. This approach is aimed at mitigating frustration and ensuring a smooth flow of the learning process.

Participants

Our study participants were 52 eleventh-grade secondary students who were 16–17 years old. All but one student had no prior experience using *GeoGebra*, even though some students had seen their teacher generating graphs to help them visualize mathematical concepts. We informed the students about the study's aims and asked for their consent to use their data to evaluate their learning experience. We assured students that we would handle their data anonymously and respect their privacy, and we took ethical considerations into account to ensure good research practices for underage populations.

Data Collection

Workbooks and *GeoGebra* Files

To understand better the feasibility of integrating CT-embedded learning activities into calculus lessons, we examined the workbooks and *GeoGebra* files of students during both the plugged (working on *GeoGebra*) and unplugged (working on workbooks) phases of the learning activity. Evaluating CT-embedded activities allows us to identify potential common mistakes in students' work. At the end of the lesson series, we collected all workbooks and *GeoGebra* files from students. We communicated to the students in advance and provided clear instructions on the procedure for returning their workbooks and uploading their *GeoGebra* files. Regarding the upload of files to our online repository, students had the option to upload individual files after each lesson or to submit all their files at the end of the series. We provided clear guidelines for both options and were available to assist with any questions or technical issues that may arise. The first two authors checked and cataloged each submission to ensure all workbooks and files were accounted for. However, students could choose not to share their files with us. Their decision was completely voluntary and did not impact their participation in any way.

Artifact-Based Interviews

Qualitative methods are an appropriate choice for examining complex and sophisticated thinking skills like CT in a comprehensive way. To better understand students' learning experiences with the designed learning activities, as well as which kind of strategies they employed to tackle the computational problems at hand, we conducted one-on-one semi-structured, artifact-based interviews (Brennan & Resnick, 2012) with them. The workbooks and GGB files were the foundation for reflection on the developed activity during the interviews. The main questions and the respective topics can be found in Table 3. The final part of the conducted interviews

Table 3 Main questions in semi-structured interviews

Question topic	Question
General	How did it go? What issues did you run into (regarding GGB, formulas, syntax, conditions, etc.)? How were you able to solve the problems that occurred (workbook, alone, peers, or teacher)? What do you think about *GeoGebra*? What advantages/disadvantages do you see? How was it to first write down your steps and then work in *GeoGebra*?
On the assignments (artifact-based questions)	How was it to define variables and functions in *GeoGebra* and use commands? (show *GeoGebra*, if–then-else, row, macro, and iteration list) How did you take the step from a specific to a general solution? (shown in *GeoGebra*) How did you evaluate when you were done with the task? (show in *GeoGebra*, exception(s)) Look back at chapter 7: How did you fill in the flow chart to describe the Newton–Raphson method? (7.1, show workbook/paper) How do you think you can use *GeoGebra* to approximate the other zero point using the Newton–Raphson method? (7.9) How did you go about approximating the golden ratio? How did you reuse your earlier solution? (7.10) How did exploring the effect of changing the starting value go? When does it go wrong? (7.11)
Reflection	What do you think of the lesson series? Any tops and tips? Can you describe what you learned in general in this lesson series and how it might help you in solving problems in the future?

included questions regarding students' reflections on their experience with *GeoGebra* and how they perceived the lesson series.

In total, 25 students participated in the interviews, which took place at the end of the lesson series. Each interview lasted between 10 and 25 min. During the semi-structured interviews, we asked students about (1) their opinions on the learning activities, (2) the difficulties they encountered, (3) how they overcame them, (4) how they implemented CT concepts and practices, and (5) their experience with CT-embedded calculus lessons.

Data Analysis

Workbooks and *GeoGebra* Files

We evaluated the CT-embedded activities on three levels similar to previous evaluations of computationally rich learning activities (Chytas et al., 2018; van Borkulo et al., 2021): "incorrect" for activities in which less than half of the tasks were correctly completed, "partly correct" for activities in which at least half of the tasks (but not all) were correctly completed, and "correct" for activities without mistakes (e.g., when the code of students accurately generated the intended graphs, such as a

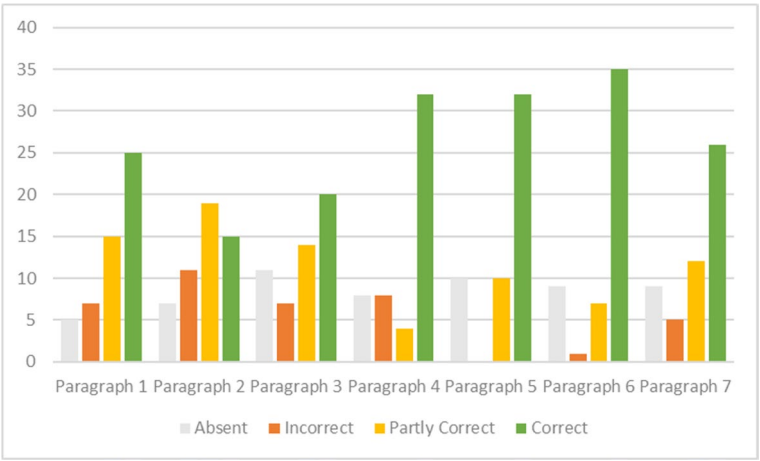


Fig. 3 Workbook analysis results (physical class)

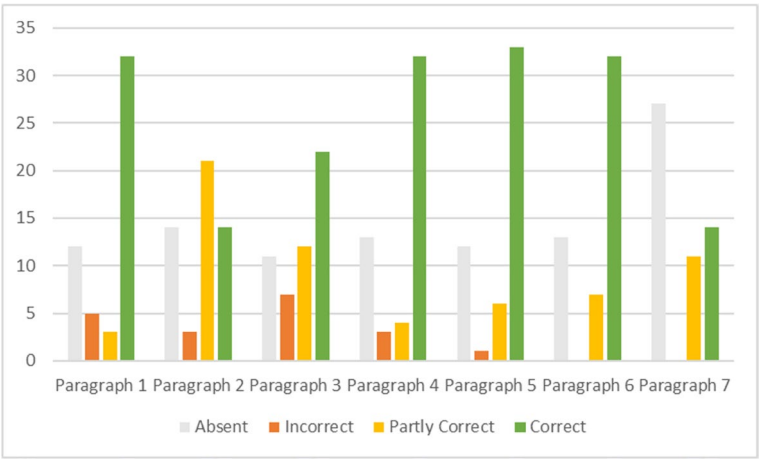


Fig. 4 *GeoGebra* file analysis results (physical class)

parabola). Additionally, we also took into account the methods used to achieve the results including mathematical accuracy and efficiency of the commands used.

Each task in the workbook was designed with clear mathematical and computational objectives, such as constructing a perpendicular line of a graphing a line between two given points. The *GeoGebra* files and Workbook analysis respected the diversity of approaches to engaging in mathematical problem solving using *GeoGebra* to explore, visualize, and analyze mathematical and CT concepts. Some students wrote code that was not needed for solving the calculus problems at hand, but we still coded their activity as "correct" if their solution worked. Some students missed classes, did not save their *GeoGebra* files, or chose not to upload them for our data

analysis. Therefore, we coded them as "absent." The seventh chapter was the most challenging one and not mandatory for the students, but only 12 files of the students/groups were absent (Figs. 3 and 4).

For the activities that were evaluated as "incorrect" and "partly correct," we also included notes regarding the mistakes of the students, which we later coded inductively. An example of such coding could be when students did not consider the special case of a general solution, e.g., in situations where the denominator of a fraction becomes zero, as it leads to an undefined result.

The assessment of the activities focused on ensuring the solutions accurately addressed the mathematical and CT concepts being tested and identifying errors that affected the accuracy of the solution and incomplete tasks. Thus, the workbooks and *GeoGebra* file analyses did not aim at evaluating the learning outcomes but at identifying common mistakes and encountered difficulties when using *GeoGebra* in CT-embedded calculus learning activities.

Interviews

The semi-structured interviews were analyzed using inductive and deductive coding approaches. Our preliminary set of deductive codes included codes from previous work (van Borkulo et al., 2021) focusing on participants' previous experience, perceptions of the learning activities, encountered problems, and computational thinking aspects, e.g., algorithmic thinking and strategies to tackle computational problems in *GeoGebra*.

Wherever possible, we triangulated the data from different sources to provide robust evidence of our findings. We used data triangulation and investigator triangulation (Carter et al., 2014) in iterative circles of design and analysis. The researchers worked in close cooperation and met weekly to compare codes and themes that emerged from the collected data.

Results

We present the study's findings with insights into how students perceive the integration of CT into calculus lessons with *GeoGebra* and what challenges they face in successfully completing the developed activities. In the next sub-sections, we take a closer look at the respective data sources one by one to provide a more comprehensive picture of the results.

Findings from the Analysis of Workbooks and *GeoGebra* Files

According to the teachers who implemented the lesson series in their classes, the learning outcomes of the lesson series were satisfactory. This is also supported by the analysis of workbooks and *GeoGebra* files, which shows that the students started getting more familiar with *GeoGebra* and computational problem-solving. The decreasing frequency of mistakes in the later chapters, despite the progressive

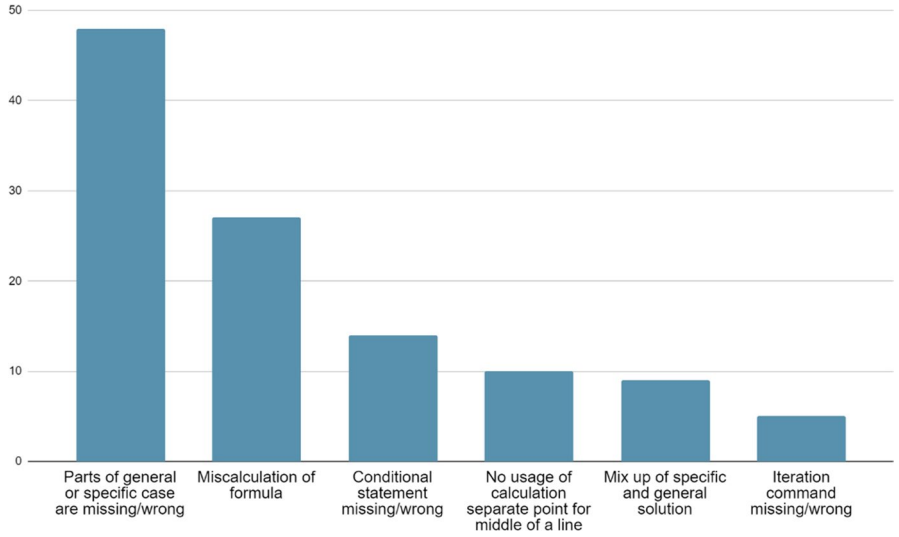


Fig. 5 Students' mistakes in workbooks

increase in the difficulty of the activities, suggests a positive development in students' learning in terms of mathematics content knowledge and use of CT skills. The later chapters introduce more sophisticated computational concepts (e.g., using iteration lists and macros) and algorithms (e.g., the Newton–Raphson method) which also require students to build more extensive and complex code in *GeoGebra*.

The most common mistakes in the workbooks were related to generalization and the students not creating a general solution for the activity (e.g., by not using variables and conditional statements to consider the special cases). For example, when calculating the slope of the original line to find the negative reciprocal for the perpendicular bisector, students might forget to check for a zero denominator, which would lead to an undefined slope. Very often, the students used the wrong formula for equations which led to wrong solutions.

For example, a student trying to create a line through two given points might mistakenly calculate the slope of the line using the wrong formula and end with a wrong line equation as a result. Considering that the problems at hand provided space for students to test their solutions, there are indications of students' mathematical content knowledge gaps. Therefore, such gaps could become barriers to moving to more sophisticated computational endeavors with *GeoGebra*. This is also illustrated in Figs. 5 and 6.

The most common mistakes in the workbooks were related to mistakes in creating general and specific solutions to the problems. Miscalculations of formulas and wrong use of computational concepts were also commonly found mistakes, using wrong conditional statements for considering special cases and using wrong conditions for the iteration lists (conditional loops). For example, in the final assignment, students might have used the wrong iteration statement not updating the x-value correctly. This could be related to students' lack of

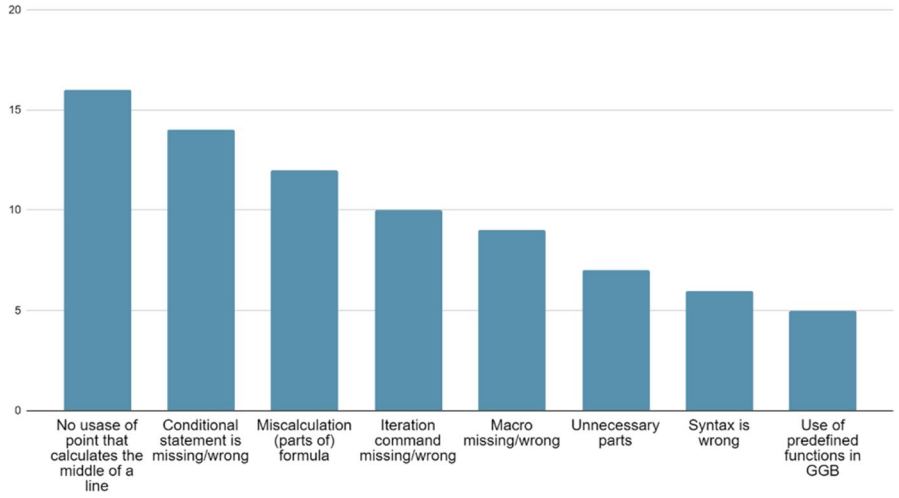


Fig. 6 Students' mistakes in *GeoGebra*

understanding of mathematical concepts being applied, e.g., the properties of a parabola or the Newton–Raphson algorithm.

Another reason could be that students did not have adequate time to familiarize themselves with computational concepts like iterations and conditional statements; in the later sessions, the *GeoGebra* file analysis shows that they developed a better understanding of such concepts. Finally, some students might have lacked attention to the details and did not evaluate their solution, as long as the generated graph in *GeoGebra* seemed functional at first look.

The students in the online class encountered more issues than in the classes that took place at school. The rates for absent files (Janssen, 2021) were significantly higher than in the classes that took place physically, and the students seemed more frustrated when working on *GeoGebra*. This finding was anticipated because the lesson series was shorter, and the teacher of the online class created online rooms for every student or pair of students. Furthermore, the teacher could not efficiently address all arising questions besides the ones in the first and last 5 min of each lesson when all students were in the same online room. The students of the online class who engaged in the lesson series made similar mistakes to the students in the physical class (Figs. 7 and 8).

Findings on Encountered Problems from Artifact-Based Interviews

In addition to the student mistakes identified in the workbooks and *GeoGebra* file analysis, students reported problems they encountered during the interviews. Below, we present some key themes that emerged.

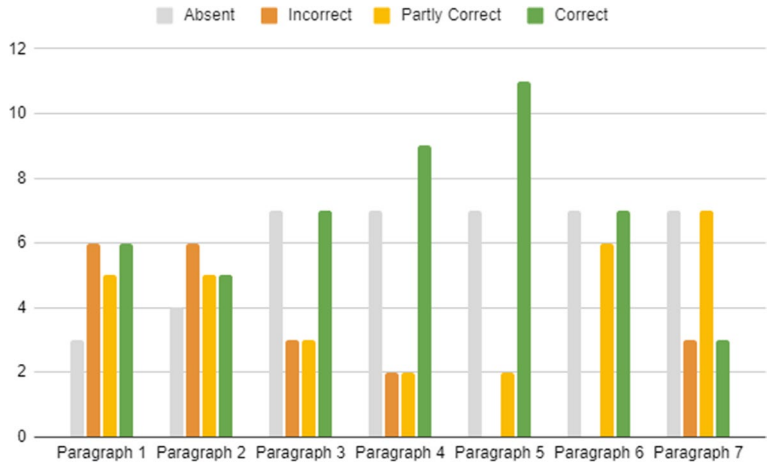


Fig. 7 Workbook file analysis (online classes)

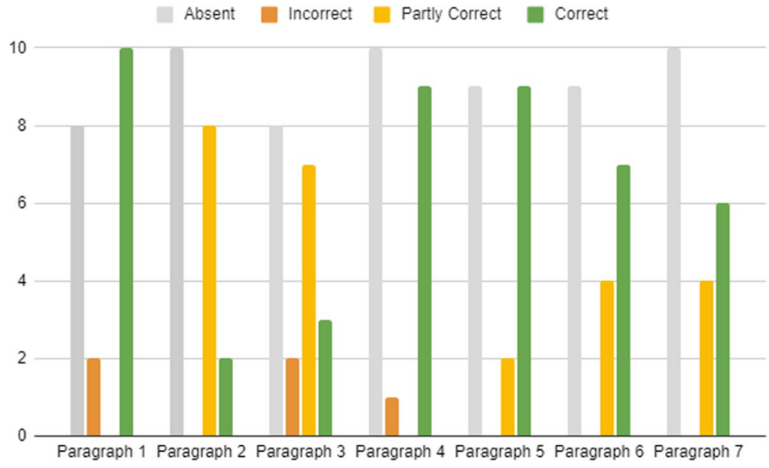


Fig. 8 *GeoGebra* file analysis (online classes)

Getting Familiar with *GeoGebra*

Eight students mentioned the initial difficulties they faced when using *GeoGebra*, which led to frustration and discouragement with using the software.

It's more the program GeoGebra […] Bit of trouble with that to fill in everything.

Getting familiar with *GeoGebra*'s interface was vital to engaging in computational problem-solving in the developed activities and using the core elements of CT as seen in Table 1 and 2. A particular frustration of students when they first used the software was that they had trouble with mathematical problems that they

could easily solve with pen and paper (e.g., drawing a tangent) and took longer times to do so with *GeoGebra*.

> Yes, for example, drawing up a tangent or something, that's not very difficult. Sometimes it did take a long time to draw up a tangent and you could actually do that very quickly, but otherwise it was just necessary.

However, after multiple attempts, students got used to the software, something that is also captured in the *GeoGebra* file analysis (despite the activities' increasing difficulty, the students made fewer mistakes in the later chapters).

> Well, in the beginning, I sometimes didn't know how to fill something in GeoGebra, but after a few tries it just worked. And the steps were just clear in that booklet.

Clear instructions and clarity in using *GeoGebra* in additional resources such as the workbooks were beneficial for the students who grew confidence and competence in using the software.

Syntax

Many students communicated with us the challenges of using *GeoGebra* and the difficulty they had filling in the information correctly. In certain cases, students showed frustration with receiving error messages and felt unsure about what they were doing because of the interface and syntax of the *GeoGebra* environment, which required time to get used.

> Well, I do find it difficult. Yes, I don't know [...] I do have trouble filling in things and also when I fill it in, I get one of those triangles saying it's wrong and then I get a bit mad, I think: well, why not? What am I doing wrong again? Because of course you have to fill in exactly what the computer wants.

> Yeah I wasn't very specific but then in GeoGebra you need to be specific so I think that that also changed a little bit if I need to do that exercise again, I will be maybe more specific with every little thing because normally you just think 'oh, that's kind of easy' say 'it is normal, you forget that' […] But in GeoGebra you need to be [...] It doesn't work that way. You need to be very specific and that also [..]. In the first exercise I really didn't do that actually, I think.

> Well, you really had to complete everything step by step because, of course, the computer doesn't immediately know everything you think how it works, so that took some getting used to […].

Overcoming syntax issues required students to debug and refine their approach to solve the calculus problems at hand, break them down into smaller sub-problems, and become more precise when using commands in the *GeoGebra* environment. Debugging *GeoGebra* code provided opportunities for understanding how algorithms work and why something might go wrong.

Gaps in Mathematical Content Knowledge

Students showed a good understanding of how to use computational concepts in *GeoGebra*, but often mathematics content knowledge gaps hindered, effectively applying them in computational problem solving in the developed activities. In many cases, students understood computational concepts in *GeoGebra* (e.g., iterations and conditional statements) to a satisfactory degree but failed successfully to complete the developed activities due to gaps in mathematics content knowledge. Without a solid foundation in the mathematics content, students were unable to use computational concepts and enter the right information into *GeoGebra*, leading to errors and wrong results. During the interviews, five students specifically referred to mathematics content knowledge gaps or the need to refresh their memory in order to recall definitions of mathematical terms, characteristics, and properties of shapes (e.g., parabolic geometry).

> At the beginning, I had to think for a moment […] okay, what is a parabola again? How do I set it up? And so on […] Just a bit of that sort of thing.
>
> Yes, setting up tangents and stuff. Of course, we knew that, but it took some time getting used to it again.
>
> Well [...] It's been a while so I'm just trying to think of what it was again." [referring to parabolas]

During the interviews, seven students mentioned they needed their teacher's support to tackle occurring problems. Eleven students mentioned that their classmates helped them with activities, and six students specifically mentioned that they were able to tackle an occurring problem by using the hints of the workbooks.

Findings on Students' Learning Experience

Most students welcomed the idea of using *GeoGebra* in CT-embedded mathematics lessons. In the interviews of students in the classes that took place physically, the 25 students mentioned positive aspects of the experiences with the lesson series while seven students mentioned negative experiences with the developed learning activity.

Positive Attitudes Toward Working with GeoGebra

A "Different" Approach to Mathematics

Seven students stated they liked the developed lesson series for different reasons, including the possibility of addressing mathematics problems with digital tools, which makes the activity more fun.

> I actually found it quite fun because normally we do the same things, and now we do something different with math [...] I've never done anything else than just in a book. Because with biology, you have bio practices, of course, but never with mathematics. So, it's nice to do something different for a change.
>
> But I think it's nice that you don't have a normal lesson with your book and your notebook. That you are doing something different, something new, so to speak. So, I like that.

Such statements indicate that students have a positive attitude toward using *GeoGebra* in their mathematics lessons, as they find it to be a fun and engaging change from traditional book-based learning. Moreover, as they have never used anything other than a book to learn mathematics in the past, they highly appreciated the opportunity to use technology to learn and enjoyed trying something new and different. Such preference of students toward different teaching approaches can be an opportunity for introducing interactive learning activities that turn abstract mathematical concepts into more tangible and attractive learning experiences.

The possibility to enable students to take a more active role in learning was also appreciated by the students. Two students stated:

> I thought it was anyway sincerely a fun series because it's more fun than just sitting in class and just listening to the teacher and doing assignments. So, it was something different and that was kind of fun.
> I liked it more than just normal mathematics, because it is more problem solving than just 'here you have the same problem, do it 100 times over [...]'. It is more thinking than doing, which I like.

Typical CT-embedded mathematics learning activities with digital tools can foster autonomy by enabling students to explore, engage in computational experimentation, and discover problem solutions independently.

Structured and Clear Learning Material

16 students connected their positive experiences with being able to complete the activities successfully and follow the workbook in a clear and structured way besides miscalculations and the need for test and debugging. One student stated:

> It's going pretty well, I would say. Most of it is finished. It just worked out. Well, with most of them I just read the assignment and I could figure most of it out. And also GeoGebra worked just fine. It was pretty clear how that worked […] And my biggest problem was [...] Sometimes I messed up a command and I couldn't figure out what went wrong for a while, but I eventually got everything.

Like with every learning activity, structured and clear learning materials are vital in CT-embedded mathematics learning activities and can be effective tools for supporting the learning process. The students appreciated the developed workbooks that

provided examples of how *GeoGebra* works, syntax hints, and connections to what was learned in previous sessions, gradually enabling students to advance to more computationally complex assignments independently in the last chapters.

Features of *GeoGebra*

The advantages students saw were related to computational aspects of *GeoGebra* enabling visualization, evaluation through testing, automating processes, being versatile and useful, and promoting different ways of working and thinking about mathematics problem-solving.

> Yes, because sort of, you don't have to draw it [the graph] yourself and it's right [...] It gives the points right away and you can move lines and then it gives the points with it and you can test your solution, that's very convenient.

> Well, GeoGebra shows it [the graph] when you fill in something and you see what happens. With mathematics, you don't normally see it in front of you and this is very useful.

Such statements indicate that students perceived the software as being useful and convenient for learning mathematics and solving mathematics problems, especially those that are difficult to visualize. The software's feature that enables plotting graphs and displaying points quickly and accurately, as well as its ability to adjust lines dynamically and testing, were perceived as major advantages. Interactive exploration and experimentation with *GeoGebra* enabled students to formulate hypotheses (through algorithm design), test them, and refine them (through debugging their code) when necessary.

A student also mentioned that, with a tool like *GeoGebra*, you can create general solutions, so that, if you make a mistake, you can revise the activities without having to start from scratch.

> The advantage is that when you get a new problem you can easily adjust the lines by just dragging the points around, which is a lot faster than just having manually to adjust everything again. So, if you are doing a similar problem multiple times, that would be easy. Yeah, just the calculating and you get to see the lines without having to draw it […].

The student saw the interactivity and ease of use of *GeoGebra* as a major advantage and noted that being able to adjust lines quickly by simply dragging points is faster and more efficient than manual adjustment. This feature makes it easier to work on similar problems repeatedly, so the student appreciated being able to calculate and visualize lines without having to draw them by hand. Hence, *GeoGebra* is perceived to be a time-saving and efficient tool for working on mathematical problems. Moreover, this feature provides opportunities for fostering pattern recognition skills. By manipulating the *GeoGebra*-generated lines, students can experiment with the underlying mathematical principles and patterns of the programs' behavior and understand them in more depth than text and oral explanations.

Negative Attitudes Toward Working with *GeoGebra*

During the reflection questions in the last part of the interviews, the following themes emerged.

Unexpected Outcomes, Slow Loading Times, and Technical Errors

Despite *GeoGebra* being a reliable and powerful tool, four students experienced technical issues such as software crashing, slow loading times, compatibility issues with their devices, and unexpected outcomes that significantly hindered their progress.

> Sometimes there was just something not coming out […] GeoGebra was loading for too long or you accidentally clicked something [...] Did you kind of have to copy something and then you clicked back, only then you can't continue again in the same formula box, because that's already been used, so then you have to fill it out all over again. So that took a lot of time.

> When you opened another thing, it didn't keep the other one that you had done before. And it didn't load sometimes. You couldn't click on it because it was maybe too much at once, so that was [annoying] [...] But the rest was okay.

Such technical issues had a negative impact on the learning experience of students and required valuable time to tackle them. Smooth, user-friendly, and stable software environments are essential in engaging students in computational problem-solving with mathematical tools, and commonly found issues should be addressed by teachers and mentioned in learning material.

Seven students saw disadvantages in using *GeoGebra* in their lessons because they sometimes do not manage to use the program as intended, they might get stuck in some parts, and they did not know how to proceed using *GeoGebra*. According to them, in contrast to working with pen and paper, technical issues like errors, long loading times in complex assignments, confusing interfaces, and teachers sometimes being unable to help students tackle technical issues that occur when working with *GeoGebra* were too frustrating.

> A disadvantage of it. Sometimes it gets stuck when you enter things [...] With that graph, the computer can't handle it [...] Especially that tangent, then it gets a little stuck.

> Well, sometimes it didn't quite do what it was supposed to do. So sometimes it wasn't very easy.

Difficulty with Troubleshooting

In some cases, students experienced frustration and difficulties when trying to find help with certain problems, and there was no one able to support them.

> As with chapters five and six, you occasionally encounter problems where you need help finding a solution. But you cannot find any help, and that is frustrating because you have the idea of what you do not understand, but no one can help you [...] and then it's up to you [...] but then it's just a tiny thing, and you are like 'oh, here I made a mistake' [...].

Consequently, students might feel frustrated and are ultimately forced to rely upon themselves to find a solution, even if the solution is fixing a small mistake. This experience highlights the importance of having access to support and resources when encountering problems in the learning process. Three students shared that they had an overall negative experience with the lesson series due to the difficulty of the mathematics content, code errors, and the complexity of the activities. These challenges might have potentially hindered their learning outcomes.

Discussion

Several studies have reported on the benefits of using *GeoGebra* in mathematics classrooms, in terms of developing mathematics content knowledge and mathematical thinking (Arbain & Shukor, 2015). The current study identified advantages and challenges in working on CT-embedded, 11th-grade calculus tasks using *GeoGebra*. The aim of the developed lessons series was to foster CT skills in calculus lessons, but issues like students' mathematics knowledge gaps and inexperience in computing created challenges in engaging them in computational problem-solving.

Previous studies with *GeoGebra* focused explicitly on students acquiring mathematics content knowledge. Our study considered the CT dimension in the lesson series which benefited from the same advantages in previous studies with *GeoGebra*, e.g., visualization, evaluation, testing, automating processes, being versatile and useful, and promoting different ways of working and thinking about mathematics and problem-solving. Visualization can be a powerful feature for comprehending mathematics (Drijvers, 2018) and CT aspects in learning activities. The results of this study showed that interactivity and visualization are particularly useful in understanding and mastering mathematics content knowledge and CT skills through computational experimentation.

Since a universally accepted definition for CT does not exist in the academic literature, our developed activities focused on specific aspects of CT that are most frequently considered as its core elements: decomposition, pattern recognition, abstraction, and algorithmic thinking/design (Dong et al., 2019; Kynigos & Grizioti, 2018). We presented how these aspects can be addressed in CT-embedded calculus activities with accessible mathematics software that tends to be popular in educational practice, in our case *GeoGebra*. *GeoGebra* is widely used by mathematics teachers in the Netherlands, and its features provide abundant opportunities to introduce CT

to secondary students. Even though many mathematics teachers in the Netherlands use *GeoGebra* in their lessons, our interviews with students showed that about one-fourth of the students take a passive role in the classroom before participating in the lesson series with the teachers using the software for demonstration purposes only. Our approach puts students in a more active role with mathematics tools and ideas and focuses on promoting CT skills in calculus lessons through computational experimentation.

We collected data during the developed lesson series from different sources including workbooks, *GeoGebra* files, and artifact-based interviews with students. The data analysis of different sources provided unique insights from the respective data and methods that allowed us to capture and evaluate the students' learning experiences and the challenges they encountered. The analysis of students' activities in *GeoGebra* and the workbooks allowed us to identify common mistakes and content knowledge gaps about mathematics content knowledge and computational concepts and practices, e.g., when students were trying to generate graphs under specific conditions—often students used false conditions, macros, and parameters due to their gap in mathematics content knowledge. In addition, during the interview phase, the students reflected on their learning experiences and problems they encountered during the lesson series.

Our findings suggest that the students participating in the online classes struggled more with the lesson series than the students attending the physical classes. Moreover, the teacher of the online class mentioned that his students were strong in both mathematics and computing, which raises concerns about the instruction approach in the online setting and time distribution for the learning activities. The developed material was designed considering the teacher as a facilitator, not an instructor, but it seems that it would be beneficial to include more scaffolding, as well as more time for computational experimentation to get familiar with *GeoGebra* and basic computational practices. Students stated they particularly struggled with learning the syntax for *GeoGebra*. They had to adapt to the software's limitations and realize that the computer did not immediately understand their intentions.

Using correct syntax when using the software can be a constructive opportunity to develop CT (Wing, 2011), as students need to be precise and define a problem and express its solution(s) in ways that *GeoGebra* can interpret. In doing so, students may need to engage in CT practices like testing and debugging (Brennan & Resnick, 2012) and approach errors systematically. Teachers should provide adequate support in addressing syntax issues with step-by-step examples, interactive demonstrations, *GeoGebra* cheat sheets, and appropriate scaffolding. Our *GeoGebra* file analysis may provide a starting point for analyzing common mistakes that students make in such activities and discussing them in the classroom. Such discussions can focus both on CT and mathematical content knowledge.

During artifact-based interviews (Brennan & Resnick, 2012), we were able to capture students' encountered challenges and understand better aspects that they disliked. Such insights are important not only for education researchers but also for teachers who strive to equip their students with digital literacies like CT. This calls for adaptations in evaluating the CT learning outcomes of students through simplified pre- and post-tests, and looking more in-depth at the learning process and CT

implementation of students, identifying and addressing the problems that inevitably occur when working with computational tools. Using different methods and data sources enabled us to triangulate our data to understand student-encountered challenges better. The findings on understanding students' common mistakes and encountered challenges in our developed activities reinforce the argument that capturing CT skills requires various assessment methods (Brennan & Resnick, 2012). In our view, artifact-based interviews and reflection reports can provide valuable insights into students' experiences and challenges in working on such activities that might not be captured using typical assessment methods due to the complexity of CT skills.

The analysis of the *GeoGebra* files also revealed that, despite the activities' increasing difficulty, the students made fewer mistakes. Our data analysis also showed that, despite students mentioning encountering difficulties in the interviews, they got used to the software and were able to address the CT-embedded calculus problems with more confidence. This is also aligned with feedback from the teachers of the classes, according to whom most students successfully engaged in the developed activities. This indicates that the designed lesson series could potentially contribute to developing pre-university students' CT skills, but more research is needed to evaluate its impact effectively.

With regard to our developed *GeoGebra* activities, we refer to generalization as part of the algorithmic design/thinking, contrary to our previous work (van Borkulo et al., 2021). That is because, in order to translate calculus problems into *GeoGebra* code, students were required to use variables, loops, sequences, and selection structures to consider special cases in the equations, which are crucial aspects of algorithmic thinking/design.

The interviews with students indicated that digital tools like *GeoGebra* are welcomed by most students, as they add a more fun and independent learning dimension to calculus lessons. The *GeoGebra* environment enables students to work more efficiently on the activities under the condition that there is adequate support and that the tasks are structured and clear. Our findings suggest that successfully completing the activities and overcoming technical difficulties is rewarding for the students. Considering mathematics content knowledge gaps of students and providing adequate resources for tackling technical issues are key to having a positive learning experience.

While our study focused on the feasibility of integrating CT into calculus lessons using *GeoGebra*, its findings have implications that can be extended to various areas within mathematics where *GeoGebra* or similar tools can be used. Computational concepts that *GeoGebra* supports, e.g., iterations, can be used to approximate solutions of equations with the Newton–Raphson method for finding roots (like in our activity), as well as exploring reflections, translations, and rotations of shapes in geometry lessons, etc. In the same way, computational concepts like conditional statements can also be used in probability simulations (among other areas) to determine outcomes, behavior, and interactions of the generated scenarios. *GeoGebra*'s features provide promising opportunities for computationally rich mathematics activities that can potentially engage students in real-world problem-solving and enhance learning through interactivity and visualization. We hope that our study

will seed ideas for creative ways to integrate CT aspects into diverse areas of mathematics education through *GeoGebra* or similar dynamic mathematics software tools.

Conclusion, Limitations, and Future Work

Addressing CT aspects in CT-embedded mathematics education is challenging due to their core similarities with mathematical thinking. In the Netherlands, computing and CT are adequately addressed only in elective computer science courses in secondary education, despite being essential for preparing students for employment and digital citizenship. Therefore, introducing CT in mandatory mathematics lessons is beneficial in addressing equity issues and promoting more opportunities for underprivileged and underrepresented students by increasing participation in computing.

Our study showed that powerful tools that are accessible to teachers could be assets in fostering CT skills in pre-university students during mathematics lessons. The students have managed to solve the computational problems in *GeoGebra* to a satisfactory degree. The didactical approach and providing sufficient time for familiarizing the students with new tools are key to addressing successfully the developed activities. Sufficient scaffolding during the lessons series is essential, and adjustments for providing enough support for students with no sufficient experience in computing and mathematics content knowledge gaps need to be considered.

There is a need for continued practice or review of mathematical content knowledge to maintain students' skills and knowledge before introducing computational concepts and practices. Additionally, it is critical to consider that understanding mathematical content knowledge today also requires understanding and using digital tools for mathematics (Geraniou & Jankvist, 2019). Successful learning in calculus using tools like *GeoGebra* depends on both mathematical and digital competences, including the ways that these tools function and understanding their capabilities and limitations.

Before discussing the implications of these conclusions for educational practice and research, we should mention the study's limitations. First, because of the COVID-19 pandemic, some students being absent in some lessons, and students not providing consent to analyze their data, our sample was smaller than expected, and the participation of students in the lesson series was not always consistent despite our efforts to provide online alternatives. Moving to online education led to a shorter lesson series and hindered student observation as teachers were overloaded with work and were limited by time constraints.

Due to time and resource constraints, we were unable directly to observe every individual student's progress with the developed CT-embedded activities. Even though our assessment method approach can provide us with insights about the successful completion rates and common mistakes of students, it has certain limitations. Direct observations of students in real-time could provide us with a more comprehensive picture of how students used the problem-solving methods they mentioned and navigated through the developed learning activities as well as the cognitive processes involved.

Performing the teaching experiments under more favorable conditions and quantitatively assessing students' learning gains would offer more constructive opportunities to investigate the learning gains of students in more depth. Even with these limitations, the *GeoGebra* files and workbook analysis, as well as the artifact-based interviews, contributed valuable insights on the feasibility of using *GeoGebra* or similar tools for integrating CT into secondary education calculus lessons and the challenges that inevitably occur.

Additionally, the study had a small scale character, which implies that its results must be interpreted and extrapolated with caution, especially because the teachers and schools involved in the study volunteered to participate. Therefore, the results may not represent the average high school in the Netherlands or Europe. More teacher professional development opportunities might be needed to equip teachers with the necessary pedagogical content knowledge and familiarize them with computational tools to embed them into their lessons. Cooperation with more experienced internal or external colleagues might be needed too.

The study's conclusions provide a basis for the topic of integrating computational and mathematical thinking into calculus lessons. There is evidence that notions of computational problem-solving form a common foundation for simultaneously addressing mathematical and computational thinking goals. Future work could investigate the possible integration of notions from the didactical theories in both computing and mathematics domains. For example, how do the notions of object formation (reification and encapsulation) commonly used in mathematics education connect to the core CT elements? Even if a start has been made to investigate this interplay, further elaboration in research is needed. Also, more research on the feasibility of the educational approach followed in this study for more scaled-up research designs, including a quantitative measurement of learning gains, would be a valuable contribution to our knowledge in the field.

Author Contribution Christos Chytas wrote the main manuscript, and together with Sylvia van Borkulo, they conducted the primary research and analysis. Christos Chytas, Sylvia van Borkulo, and Paul Drijvers met weekly to discuss findings from the analysis, which helped to refine the research and strengthen its conclusions. Sylvia van Borkulo, Paul Drijvers, Erik Barendsen, and Jos Tolboom provided critical feedback on the manuscript, contributing to the overall quality and accuracy of the paper. The contributions of all authors were vital to deliver this work.

Funding The research reported here was part of a larger project (partly) financed by the Netherlands Initiative for Education Research (project number 00517751).

Data Availability The data from this study is not publicly available but can be accessed upon request from Christos Chytas. All data is stored in accordance with the Data Management Protocol of the Freudenthal Institute of Utrecht University, The Netherlands.

Declarations

Competing interests The authors declare no competing interests.

Conflict of Interest The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Adelabu, F., Makgato, M., & Ramaligela, M. (2019). Enhancing learners' geometric thinking using dynamic geometry computer software. *Journal of Technical Education and Training, 11*(1), 44–53.

Arbain, N., & Shukor, N. (2015). The effects of GeoGebra on student achievement. *Procedia: Social and Behavioral Sciences, 172*, 208–214.

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics A study on age and gender relevant differences. *Robotics and Autonomous Systems, 75*, 661–670.

Bell, T., & Lodi, M. (2019). Constructing computational thinking without using computers. *Constructivist Foundations, 14*(3), 342–351.

Brahier, D., Leinwand, S., & Huinker, D. (2014). Principles to actions: Mathematics programs as the core for student learning. *Mathematics Teacher, 107*(9), 656–658.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of theAmerican Educational Research Association* (pp. 1–25). Vancouver, Canada.

Caeli, E., & Yadav, A. (2020). Unplugged approaches to computational thinking A historical perspective. *TechTrends: Linking Research and Practice to Improve Learning, 64*(1), 29–36.

Calao, L., Moreno-León, J., Correa, H., & Robles, G. (2015). Developing mathematical thinking with Scratch: An experiment with 6th grade students. In G. Conole, T. Klobučar, C. Rensing, J. Konert, & E. Lavoué (Eds.), *Design for teaching and learning in a networked world: 10th European conference on technology enhanced learning* (pp. 17–27). Springer.

Carter, N., Bryant-Lukosius, D., DiCenso, A., Blythe, J., & Neville, A. (2014). The use of triangulation in qualitative research. *Oncology Nursing Forum, 41*(5), 545–547.

Chytas, C., Diethelm, I., & Tsilingiris, A. (2018). Learning programming through design: An analysis of parametric design projects in digital fabrication labs and an online makerspace. *2018 IEEE Global Engineering Education Conference (EDUCON) (pp. 1978–1987)*. IEEE.

Dagienė, V., & Sentance, S. (2016). It's computational thinking! Bebras tasks in the curriculum. In A. Brodnik & F. Tort (Eds.), *Informatics in schools – Improvements on informatics knowledge and perception: 9th international conference on informatics in schools – Situation, evolution and perspectives* (pp. 28–39). Springer.

Dong, Y., Cateté, V., Jocius, R., Lytle, N., Barnes, T., Albert, J., Joshi, D., Robinson, R., & Andrews, A. (2019). PRADA: A practical model for integrating computational thinking in K–12 education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 906–912).

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

Futschek, G. (2006). Algorithmic thinking: The key for understanding computer science. In R. Mittermeir (Ed.), *Informatics education – The bridge between using and understanding computers: International conference on informatics in secondary schools – Evolution and perspectives* (pp. 159–168). Springer.

Geraniou, E., & Jankvist, U. (2019). Towards a definition of "mathematical digital competency." *Educational Studies in Mathematics, 102*(1), 29–45.

Gravemeijer, K., & Cobb, P. (2006). Design research from a learning design perspective. In J. van den Akker, K. Gravemeijer, S. McKenney, & N. Nieveen (Eds.), *Educational design research* (pp. 29–63). Routledge.

Guzdial, M. (2008). Education: Paving the way for computational thinking. *Communications of the ACM, 51*(8), 25–27.

Henderson, P., Cortina, T., & Wing, J. (2007). Computational thinking. SIGCSE. *Bulletin, 39*(1), 195–196.

Janssen, B. (2021). Incorporating computational thinking in calculus lessons: A characterisation of algorithmic thinking and generalisation skills [Unpublished bachelor's thesis]. Radboud University.

Jenkins, J. T., Jerkins, J. A., & Stenger, C. (2012). A plan for immediate immersion of computational thinking into the high school math classroom through a partnership with the Alabama math, science, and technology initiative. In *Proceedings of the Annual Southeast Conference*(pp. 148–152).

Kallia, M., van Borkulo, S., Drijvers, P., Barendsen, E., & Tolboom, J. (2021). Characterising computational thinking in mathematics education: A literature-informed Delphi study. *Research in Mathematics Education, 23*(2), 159–187.

Kynigos, C., & Grizioti, M. (2018). Programming approaches to computational thinking: Integrating turtle geometry, dynamic manipulation and 3D space. *Informatics in Education, 17*(2), 321–340.

Lee, I., Martin, F., & Apone, K. (2014). Integrating computational thinking across the K–8 curriculum. *ACM Inroads, 5*(4), 64–71.

Lv, L., Zhong, B., & Liu, X. (2023). A literature review on the empirical studies of the integration of mathematics and computational thinking. *Education and Information Technologies, 28*(7), 8171–8193.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Perković, L., Settle, A., Hwang, S., & Jones, J. (2010). A framework for computational thinking across the curriculum. In *Proceedings of the 2010 Conference on innovation and technology in computer science education*(pp. 123–127).

Rich, K., Spaepen, E., Strickland, C., & Moran, C. (2020). Synergies and differences in mathematical and computational thinking: Implications for integrated instruction. *Interactive Learning Environments, 28*(3), 272–283.

Rich, P., Egan, G., & Ellsworth, J. (2019). A framework for decomposition in computational thinking. In *Proceedings of the 2019 ACM conference on innovation and technology in computer science education (ITiCSE '19)* (pp. 416–421). Association for Computing Machinery.

Sanford, J., & Naidu, J. (2016). Computational thinking concepts for grade school. *Contemporary Issues in Education Research, 9*(1), 23–32.

Selby, C., & Woollard, J. (2013). Computational thinking: The developing definition. In *Proceedings of the 45th ACM technical symposium on computer science education, SIGCSE 2014*. ACM.

van Borkulo, S., Chytas, C., Drijvers, P., Barendsen, E., & Tolboom, J. (2021). Computational Thinking in the Mathematics Classroom: Fostering Algorithmic Thinking and Generalization Skills Using Dynamic Mathematics Software. In *Proceedings of the 16th Workshop in Primary and Secondary Computing Education (WiPSCE '21), Article 19*(pp. 1–9). Association for Computing Machinery.

van Borkulo, S., Chytas, C., Drijvers, P., Barendsen, E., & Tolboom, J. (2023). Spreadsheets in secondary school statistics education: Using authentic data for computational thinking. *Digital Experiences in Mathematics Education, 9*(3), 420–443.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.

Wilensky, U., Brady, C., & Horn, M. (2014). Fostering computational literacy in science classrooms. *Communications of the ACM, 57*(8), 24–28.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Wing, J. (2011). Research notebook: Computational thinking – What and why? *The Link Magazine, 6*, 20–23.

Wing, J. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology, 25*(2), 7–14.

Yadav, A., Connolly, C., Berges, M., Chytas, C., Franklin, C., Hijón-Neira, R., Macann, V., Margulieux, L., Ottenbreit-Leftwich, A., & Warner, J. R. (2022). A review of international models of computer science teacher education. In *Proceedings of the 2022 working group reports on innovation and technology in computer science education*(pp. 65–93).

Yazan, B. (2015). Three approaches to case study methods in education: Yin, Merriam, and Stake. *The Qualitative Report, 20*(2), 134–152.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Christos Chytas[1,2] · Sylvia Patricia van Borkulo[1] · Paul Drijvers[1] · Erik Barendsen[2,3] · Jos L. J. Tolboom[4]

✉ Christos Chytas
c.chytas@uu.nl

Sylvia Patricia van Borkulo
s.vanborkulo@uu.nl

Paul Drijvers
p.drijvers@uu.nl

Erik Barendsen
erik.barendsen@ru.nl

Jos L. J. Tolboom
j.tolboom@slo.nl

1 Utrecht University, Utrecht, The Netherlands

2 Radboud University, Nijmegen, The Netherlands

3 Open University of the Netherlands, Heerlen, The Netherlands

4 SLO, Netherlands Institute for Curriculum Development, Amersfoort, The Netherlands