Integrated Programming and Mathematics in Schools - A Solid Foundation for a Future Engineering Education?

Runar L. Berge[1], **Bjørnar Sæterås & Andreas Brandsæter**
Volda University College
Volda, Norway
0000-0002-8490-2284, 0000-0002-3217-9944 & 0000-0001-5142-854X

Conference Key Areas: *Mathematics at the heart of Engineering; Attractiveness of Engineering.*
Keywords: *Programming; mathematics; secondary education; task analysis; computational thinking.*

ABSTRACT

The interest in programming in schools has the last decade increased, and many countries have introduced programming as part of the school curriculum. Teaching of programming to students in primary and secondary school is often focused on the computer sciences aspect of programming. The current study is a part of the recently initiated research project “Programming for understanding mathematics” which has a different emphasis; the project investigates how the mathematical competence of the students are affected by actively using programming in mathematics lessons. In this paper, a recognized analytical framework for analysing the cognitive demand of mathematical tasks is presented. We extend the framework to include the analysis of tasks that utilize programming, allowing us to distinguish between tasks that are demanding due to the mathematical content, but the programming aspect of the task is trivial, and tasks that are cognitive demanding due to complex programming, but the mathematics is simple. We use the extended framework to analyse tasks in four mathematics textbooks written for 16-17 year old students by two major publishers in Norway. The results show that the tasks provided in the textbooks mainly focus on elementary programming skills, and the tasks give limited experiences with cognitive demanding programming tasks.

[1] *Corresponding Author*

R. L. Berge

runar.lie.berge@hivolda.no

doi:10.5821/conference-9788412322262.1394

1 INTRODUCTION

We have entered the era of computation. High speed calculations have become such an integral part of our life that we do not even notice; path finding algorithms show us the direction on our phones, assisted driving increases the safety on the roads, and the weather forecast allows us to know if it will rain next weekend. Behind these features, are complex mathematical models and algorithms that are solved by computers. In the professional world, especially in Science, Technology, Engineering, and Mathematics (STEM), this can be seen as a rapid change in the use of computers and computing over the last decades.

As a response to this shift in technology, programming has been introduced in the school curriculums of most European countries [1]. Most countries that introduce programming in schools argue that programming enhances logical skills and problem-solving skills [1]. For mathematics, programming is often claimed to enhance computational thinking, which can be defined as a thought process and problem-solving approach that can provide the means to translate problems into formulations that can be solved computationally (either by a computer or human) [2]. The set of skills learned from computational thinking can be promoted when tackling difficult problems. It includes decomposition (breaking a problem into smaller parts), pattern recognition (analyse and find connections in data), abstraction (identify relevant information and eliminate the extraneous details), and algorithmic thinking (develop step-by-step processes) [3], which are concepts also related to programming.

The introduction of programming in schools will change the competence of students starting at engineering education. To take advantage of the changes, it is necessary to understand how these changes are in practice implemented in schools. In this paper, we present initial results from the recently initiated research project “Programming for understanding mathematics”. The project focuses on programming at primary through secondary school, and the aim of the project is to investigate how programming can be used to enhance mathematical understanding.

While programming is introduced in schools on a higher level based on developing the computational thinking and 21st century skills of students [1], it is interesting to see how these new curriculum changes are implemented in practice. Thus, in this paper we take a closer look at how programming is introduced in the Norwegian upper-secondary school. There are mainly two different approaches that are taken when programming is introduced in schools. In many countries, programming is either introduced as a separate subject or part of an information technology course [1]. However, increasingly programming is introduced as part of traditional school courses, usually mathematics. In Norway the second approach is taken; programming is treated in several subjects, but mathematics has been given a key role in developing the students’ competence in programming. Similar tendencies can be seen in other Scandinavian countries, see e.g., [4] for an overview.

From a long tradition in mathematics, the textbook is important as a curriculum resource for teachers [5]. The tasks and how they are presented in the textbooks impact how teachers and students work with the subject. When introducing a new topic in mathematics lessons, which many teachers also are feeling unconfident about, we suppose that the textbooks may be even more important, and the content of the textbooks lay the foundation for the students' outcome of mathematics courses. Although tasks can be used differently by teachers, the way it is presented in textbooks are anyhow important for how challenging the tasks will be.

2 METHODOLOGY

In this paper we analyse how programming is treated in two different Norwegian textbook series, Mønster [6,7] and Sinus [8,9], for upper-secondary mathematics. The textbooks cover the mathematics courses of 11th and 12th grade students (age 16-17) aiming towards STEM programs at university.

2.1 Analytical framework

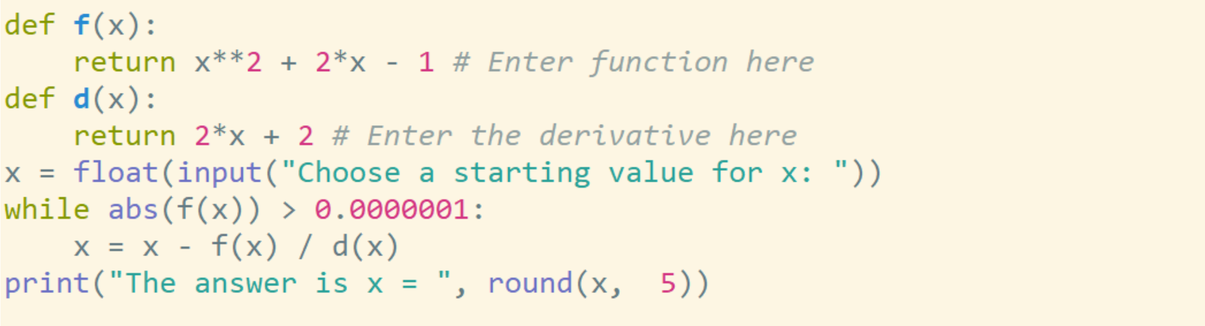
Boston & Smith [10] present a framework for analysing mathematical tasks. Tasks are assigned a number 0-4 based on the mathematical potential of the task. Level 0 is a task that does not contain mathematics, level 1 or 2 is given to a task that requires the execution of known routines or procedures while level 3 or 4 is given to highly cognitive demanding tasks that require the student to do mathematics.

In this work, we are interested in analysing tasks that include both mathematics and programming. A task can be cognitively demanding both in terms of the programming aspect and the mathematical aspect. Another task can be demanding due to the mathematical content, but the programming aspect of the task is trivial, e.g., running provided programs. Yet another task can be cognitive demanding due to complex programming, but the mathematics is simple. Thus, we extend the framework of Boston & Smith [10] to include a second axis that specify the programming potential of the task, shown on the right side of the table below. Note that a task will in general be given different levels in mathematics and programming.

Level	Mathematics (see, [10])	Programming
4	The task asks students to engage in the disciplinary activities of explanation, justification, and generalization or to use procedures to solve tasks that are somewhat open-ended in nature.	The task has the potential for the students to engage in programming. Solving the task requires an iterative process that is not predictable to the students. Building a solution must be done in a stepwise and cyclic manner with prototyping, and testing.
3	The task requires students to make connections to underlying mathematical ideas but does not	The task has the potential to challenge the thinking of students or to engage the students in making connections

	include explicit requests for generalization or justification.	between programming concepts or procedures. The student must combine several concepts in programming to solve the problem.
2	The task requires students to perform relatively routine procedures without making connections to the underlying mathematical ideas.	The task is limited to engaging students in known procedures, either specifically called for or known from prior knowledge. There is little to no ambiguity about what needs to be done or how to do it.
1	The task requires only memorization or the reproduction of facts.	The task is limited to engaging students in memorizing simple concepts or syntax. There is no need for the students to understand or make connections between the implemented code and facts.
0	There is no mathematics in the task.	The task requires no programming skills. This includes running code without the need to understand the code.

As an illustration of a typical task that is categorized to level 1 or 2 in programming, Chapter 6 in [8] gives an example of solving an equation with Newton's method (all example tasks are translated to English):



```
def f(x):
    return x**2 + 2*x - 1 # Enter function here
def d(x):
    return 2*x + 2 # Enter the derivative here
x = float(input("Choose a starting value for x: "))
while abs(f(x)) > 0.0000001:
    x = x - f(x) / d(x)
print("The answer is x = ", round(x,  5))
```

In this chapter the textbook proceeds to give 6 tasks that ask the student to solve different equations with the Newton method, e.g.,

Find approximations of the two zeros of the function:

$$f(x) = x^2 - 5x - 5$$

by using Newton's method and Python.

These tasks are therefore classified as level 1 in programming because the student should only replace the expression for the function and its derivative. It is classified to level 2 in mathematics because the student must use known procedures to calculate the derivative of the function. Similar tasks are classified to level 2 in programming if the modification to the provided program is more demanding.

An example of a task that is classified to level 3 in programming is given by task 4.92 in [6]:

> Let f be given by $f(x) = \frac{1}{5}x^2 - x - 1$
>
> Write a program that finds the smallest whole number, $n,$ such that $f(n)$ and $f(0)$ have different signs.

Here the student should recognize that the task can be solved by a combination of a while loop and an if-statement. The complexity of a script that solves this task is similar to the Newton problem above, however, in this task no examples of similar worked tasks are provided. The task is categorized to level 2 in mathematics.

2.2 Textbook analysis

First, two of the authors read all tasks given in the textbooks and sorted out the tasks that explicitly asks to be solved by programming. There are other tasks that have the potential to be solved using programming, but we only included tasks explicitly asking the student to use programming. Further, we did not include tasks that are to be solved by other digital tools, such as Computer Algebra System or graphical computer software (e.g., GeoGebra). Both textbook series use Python as a programming language. The 11th grade textbook of Mønster [6] has a dedicated appendix that is an introduction to Python. Tasks in this appendix were not included in the analysis as it mainly focuses on basic programming skills and not mathematics.

The programming tasks were then classified according to the five levels of cognitive demand along the two axes of mathematics and programming. All tasks were classified independently by two authors that have different background and competence; one author has a background from engineering and applied mathematics with considerable knowledge about programming, and the other author has extensive experience from teaching of mathematics in upper-secondary education, with less knowledge about programming. Tasks that were classified to different levels by the authors were discussed until an agreement was reached.

3 RESULTS

Table 1 shows the number of tasks categorized for the two textbook series at 11th and 12th grade (16-17 year old students). In both series the number of programming tasks is more than doubled in the second book (12th grade) compared to the first book (11th grade). Furthermore, programming related tasks cover a wide range of mathematical topics and are given in almost all book chapters of both series. Fig. 1 shows how the programming tasks are classified for the 11th and 12th grade textbooks. The average programming level is 1.9 and 1.8 in the 11th and 12th grade textbooks, respectively, and the average mathematics level is 1.6 and 1.9 in 11th and 12th grade textbooks, respectively.

Table 1. The number of tasks in the four textbooks that include programming. The parentheses give the ratio of programming tasks to the total number of tasks for each textbook.

Name of textbook series	*11th grade*	*12th grade*
Sinus	20 (1.7 %)	46 (4.5 %)
Mønster	24 (2.2 %)	60 (7.9 %)

Table 2 shows how all tasks in the two textbook series are classified. 105 of the 156 tasks are classified as having a low cognitive demand (level 2 or lower) in both mathematics and programming. There are 35 tasks that are classified to level 3 or 4 in mathematics and 26 tasks that are classified to level 3 or 4 programming.

Even though the majority of tasks are classified as less demanding (level 0-2), many of these tasks do include relatively complex programming and mathematics. The reason they are given a lower level is that examples shown previously in the text are very similar to the tasks given the student. Both textbook series extensively give tasks that require the student to modify provided examples. These tasks are mainly classified to level 1 or 2, depending on the complexity of the modifications.

We will now study two selected tasks in depth. Task 1.32 in Mønster [7] is translated as follows:

> The sum below converges to Euler's number e quickly:
>
> $$1+\frac{1}{1}+\frac{1}{1\cdot 2}+\frac{1}{1\cdot 2\cdot 3}+\frac{1}{1\cdot 2\cdot 3\cdot 4}+\cdots$$
>
> Write a program that asks the user for the number of terms in the sum and then calculate the sum of these first terms.

This task can be solved by a double loop (or a single loop and using the factorial function in the Python *math* library) and the students are not given any similar examples. The task is classified to level 3 in programming, and in terms of mathematics it is classified as level 2. The other textbook series, Sinus [9], gives an equivalent task (1.306) where the students are asked to study different series

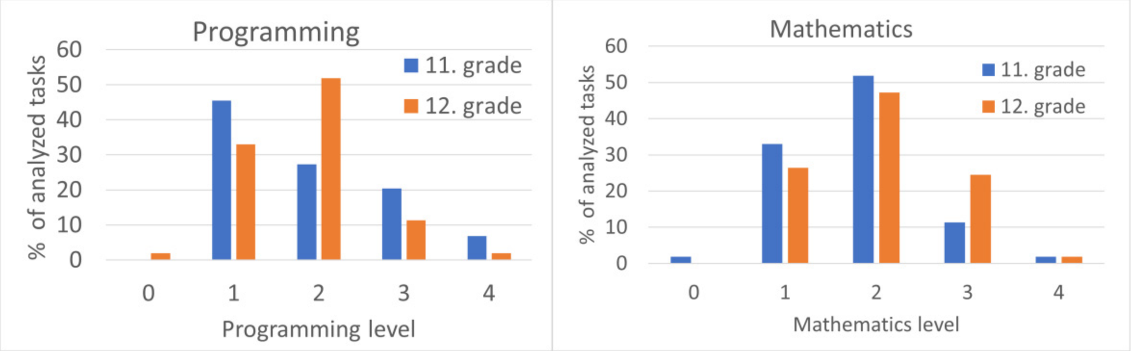
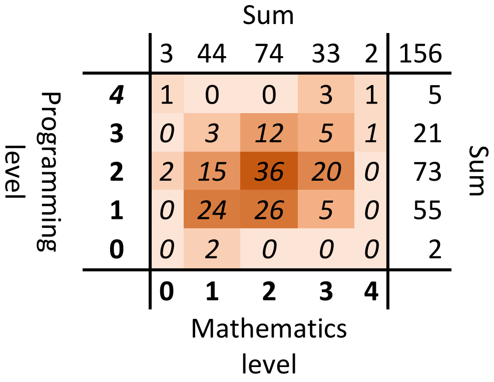


Fig. 1. The fraction of programming tasks at each cognitive level for programming and mathematics.

Table 2. The number of tasks classified to the different levels for programming and mathematics. E.g., there are 36 tasks classified to level 2 in both programming and mathematics and 20 tasks that are classified to level 2 in programming and level 3 in mathematics. The top row and rightmost column give the column sum and row sum, respectively.



Programming level \ Mathematics level	0	1	2	3	4	Sum
Sum	3	44	74	33	2	156
4	1	0	0	3	1	5
3	*0*	*3*	*12*	*5*	*1*	21
2	*2*	*15*	*36*	*20*	*0*	73
1	*0*	*24*	*26*	*5*	*0*	55
0	*0*	2	*0*	*0*	*0*	2

expansions of the Euler number. However, this textbook takes a different approach; Here the task provides the students with a script for calculating the sum given in the task above. The students are then asked to modify the program by trying different approximations of e, e.g.,

$$e = \frac{1}{2}\left(\frac{1}{0!} + \frac{2}{1!} + \frac{3}{2!} + \frac{4}{3!} + \frac{5}{4!} + \cdots\right),$$

and compare the different approximations and reflect, discuss, and evaluate the different convergence rates of the sums. This task is classified as level 2 in programming and level 3 in mathematics. This illustrates how different formulations of a task may change the classification of the task. In Mønster, the focus is on the implementation of the task in Python, while Sinus uses programming as a tool for studying and understanding different convergent sums.

Both textbook series use programming to motivate concepts in mathematics. In task 2.72 in [9], programming is used to investigate the derivative of the natural logarithm. Both derivatives and numerical approximations (including Python implementation) are presented previously in the chapter, however, the analytical derivative of $\ln x$ is unknown at this point:

> We will now look at the function:
>
> $$f(x) = \ln(x)$$
>
> Make a program in Python that prints out a and a numerical approximation of $f'(a)$ for $a = 1, a = 2, a = 3, \dots, a = 10$.
>
> Can you generalize a rule that seem to be true?

The purpose of this task is to let the student discover the relation between the natural logarithm and its derivative, and the students must themselves make conjectures. This task is given a programming level of 2 because the students have previously been exposed to numerical derivatives and making tables of function values using a loop. The task is classified to level 3 in mathematics.

4 DISCUSSION AND CONCLUDING REMARKS

One of the main arguments for introducing programming in mathematics education is to improve the computational thinking and problem-solving skills of students. From the analysis of the textbooks in Section 3, however, we show that most of the given tasks do not require a higher cognitive demand of the students. Only occasionally do the tasks ask the student to explore, investigate or evaluate the answer obtained by programming. Of course, not all tasks in a textbook should require a higher cognitive demand, and drill and routine tasks should also be provided in a textbook. The optimal ratio of cognitive demanding tasks and practice tasks is an open area of study. A line of further research is to investigate if tasks that include programming are in general scored at a lower or higher level than the tasks that do not include programming.

In our analysis we use the presented two-dimensional framework on the programming tasks provided in four mathematics textbooks. We find the framework useful when discussing these tasks, and the framework works as a communication bridge between the two authors with substantial different programming and mathematics background. Labelling the tasks also initiated reflections upon how tasks could be improved and extended. While the framework is tested on upper-secondary textbooks in this paper, we believe that it can also be used in both higher and lower educations as well.

When classifying the tasks in textbooks we assume the students to follow the progression of the textbook. In a classroom, the teacher can use the textbooks in different ways, e.g., carefully selecting or adapting textbook tasks. This will, however, require teachers that are confident in their programming skills. The original framework in [10] has been used as a tool for selecting and adapting mathematical tasks. We believe that the extension presented in this paper can be used to select and adapt mathematical tasks that include programming. Further research is planned in the “Programming for understanding mathematics” project to study how these textbooks are used in practice by teachers.

For the engineering educations, the introduction of programming in schools will have an impact on the background of new students. As shown in this paper, Norwegian students will probably learn elementary programming with Python, but they will have limited experiences with cognitive demanding programming tasks, also on the highest level of mathematics courses in school. In most countries, programming is a new topic in the school curriculum. If the aim is to give students problem solving skills, in mathematics, or in programming, this paper shows that the curriculum resources given the teacher may be insufficient to stimulate this. For the engineering educations, it is therefore necessary to investigate both the curriculum changes and the curriculum resources such as textbooks, to understand how programming is implemented, in the respective countries, and how this might affect student outcome. Based on our experience, the proposed framework can be used as a tool to analyse tasks given in upper-secondary education.

REFERENCES

[1] Balanskat, A. and Engelhardt, K. (2015), Computing our future: Computer programming and coding - Priorities, school curricula and initiatives across Europe, *European Schoolnet,* Brussels.

[2] Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L. and Wilensky, U. (2016), Defining Computational Thinking for Mathematics and Science Classrooms, *J. Sci. Educ. Technol.*, Vol. 25, No. 1, pp. 127–147.

[3] Kaufmann, O. T. and Stenseth, B. (2021), Programming in mathematics education, *Int. J. Math. Educ. Sci. Technol.*, Vol. 52, No. 7, pp. 1029–1048.

[4] Bocconi, S., Chioccariello, A. and Earp, J. (2018), The Nordic Approach To Introducing Computational Thinking and Programming in Compulsory Education, [Online]. Available: doi: 10.17471/54007.

[5] Remillard, J. T. (2005), Examining Key Concepts in Research on Teachers' Use of Mathematics Curricula, *Rev. Educ. Res.*, Vol. 75, No. 2, pp. 211–246.

[6] Kalvø, T., Opdahl, J. C. L., Skrindo, K. and Weider, Ø. J. (2020), Mønster 1T, Gyldendal.

[7] Kalvø, T., Opdahl, J. C. L., Skrindo, K. and Weider, Ø. J. (2021), Mønster R1, Gyldendal.

[8] Oldervoll, T., Svorstøl, O., Vestergaard, B., Gustafsson, E., Osnes, E. R., Jacobsen, R. B. and Pedersen, T. A. (2021), Sinus 1T, Cappelen Damm.

[9] Oldervoll, T., Svorstøl, O., Gustafsson, E. and Jacobsen, R. B. (2021), Sinus R1, Cappelen Damm.

[10] Boston, M. D. and Smith, M. S. (2009), Transforming Secondary Mathematics Teaching: Increasing the Cognitive Demands of Instructional Tasks Used in Teachers' Classrooms, *J. Res. Math. Educ.*, Vol. 40, No. 2, pp. 119–156.