Chapter 10

Addressing a curriculum alignment problem in a Mathematical Programming course: insights from a PGDip (HE) graduate

Marcellin Atemkeng
Department of Mathematics
Rhodes University
Makhanda, South Africa

Introduction

Pursuing a Postgraduate Diploma in Higher Education (PGDip (HE)) has influenced my teaching career. What I learned on the course has had a significant impact on how I approach curriculum development, teaching and learning and assessment. It has transformed my lecture delivery, examination paper design, and overall perspective on student learning. The PGDip programme has provided me with a robust set of pedagogical tools, a deep understanding of educational theory, and valuable practical insights, all of which have been instrumental in advancing my teaching and learning practices. One significant transformation in my teaching practice post-PGDip has been the adoption of a more interactive and learner-centred approach. Now, at the end of each lecture, I introduce the forthcoming topic and encourage students to read and prepare questions in advance. This strategy not only makes better use of lecture time for dynamic discussions and problem-solving activities but also cultivates a more participatory classroom environment. It promotes a culture of inquiry, keeping students actively involved and eager to learn. Moving beyond

the traditional model of content delivery and course completion, my focus has shifted to deeply understanding student needs. I have grown more patient and open to student inquiries, showing greater sensitivity to their perspectives on various subjects.

My primary goal is to enrich students' educational journey and to adapt to their unique learning requirements. In addition to using simulation software, I have incorporated a variety of technological tools to enhance the teaching and learning of mathematics. For instance, I advocate for the use of interactive online platforms that enable students to interact with mathematical concepts in an engaging manner. These platforms provide simulations, interactive tutorials, and immersive virtual experiences that assist students in visualising and understanding abstract concepts more concretely.

I have enriched my lessons with multimedia resources, including videos, animations, and interactive demonstrations. These resources offer multiple layers of explanation and serve to reinforce learning. For example, to demonstrate geometric transformations, I employ a three-dimensional plot that allows students to observe the behaviour of mathematical functions when various variables are altered. This approach not only clarifies the concepts but also makes the learning process more interactive and impactful.

I have redesigned my assessments to align with the specific learning objectives of the course. This approach moves beyond merely testing students on isolated mathematical equations; instead, it structures assessments to reflect the learning objectives. For instance, rather than having students simply memorise and recall formulas, I craft questions that challenge them to apply these formulas in problem-solving scenarios. Understanding the significance of clear guidance on assessment criteria and expectations, I now ensure that I provide students with detailed information regarding the course sections that will be assessed at the end of the term. This strategy aids students in identifying key areas of focus and comprehending the metrics or criteria by which their performance will be gauged. The PGDip course deserves credit for providing me with a new set of

skills that have greatly enriched the practices that I discuss in this chapter.

This chapter focuses on the design philosophy of the Mathematics Department's undergraduate curriculum, in the context of fostering knowledge transfer and cumulative knowledge building for the second-year mathematical programming (MAM 201) course that I teach. I then focus on the lack of constructive alignment of the mathematical programming curriculum, on the method that I used to teach this course and the evaluation of the method with data collected from students' anonymous responses to an online survey at the end of the course. I provide examples of students' feedback that pointed to lacunae in my teaching methods, and how I used student comments to strengthen the next iterations of the course. To overcome the lacunae, I developed a strategy for teaching this course such that there is alignment between teaching objectives, assessment, and activities. I used this strategy in 2022 to teach the course, and data was collected via an online survey to access my students' responses to the method. The data was analysed and the method points to a promising way to teach the course but requires the Department to put in place additional resources to support the method and our students. I present student assessment results over a three-year period (i.e. 2018, 2019, and 2022) that indicate great improvement in student grades using the new method in 2022. Note that results for the years 2020 and 2021 are not available since it was decided to discontinue the course during the coronavirus disease 2019 (COVID-19) pandemic, the reasons for which are not pertinent to the discussion.

Reflecting on the mathematical programming curriculum

From a pedagogical perspective, a curriculum is the set of processes starting from the design, structuring and implementation of teaching and learning activities. Curriculum brings together the notion of learning objectives, course content, and planned activities for teaching and learning associated with the different ways that would be used to assess students (Young, 2014). A curriculum has a fundamental importance in the

context of higher education. It constitutes a teaching, learning, and assessment guide and it is also a conceptual tool capable of shedding light on major educational innovations (Hassard & Dias, 2013). In all cases, teaching and learning are part of the curriculum and it provides a route through which a lecturer organises the knowledge that they aim to transmit to students. The curriculum defines the "what" - that is, what learners must know and understand to solve problems in each mathematical subject and at every academic level (Niss, 2003). This approach of curriculum design is similar to two philosophical approaches of curriculum design that Toohey (1999) referred to as the performance or systems-based approach and the cognitive approach. These approaches define the aims (or goals) of the course and the lecturer teaches students towards these goals with an emphasis on building students' cognitive abilities such as logical thinking as well as cultivating a research orientation to learning mathematics. The implementation of the "what" is performed by lecturers who use their professional judgement to determine the "how" - that is, how students will achieve the goals of the curriculum through the teaching and learning processes. Mathematics is a field of abstract knowledge built on logical reasoning about concepts such as numbers, figures, patterns and transformations (Dubinsky & McDonald, 2002). Mathematics is distinguished from other sciences by a relation to reality. It is purely intellectual in nature, based on axioms declared to be true or on postulates provisionally accepted. A mathematical statement is generally referred to as a theorem, proposition, lemma, fact or corollary. Each of these statements is considered valid when the formal discourse which establishes its truth respects a certain rational structure called *proof, or logical deductive reasoning (*Stylianides & Stylianides, 2008*)*.

The establishment of links between mathematical concepts and students' daily lives, and the strengthening of problem-solving skills and metacognitive capacities, are strategies which are known to be very effective in alleviating the frustration, undermined motivation and withdrawal from mathematics that students too often experience (Presmeg, 2002). From a theoretical rather than a practical point of view, the establishment of advances in mathematics can be accomplished by defining such

practices as an adequate philosophical understanding on the part of the lecturer as to how mathematics should be taught. For example, the identification of and an emphasis in the curriculum on concepts and procedures that tend to be bottlenecks to student learning of mathematics (Middendorf & Pace, 2004) and threshold concepts (Meyer & Land, 2005), the use of objects that can be manipulated or physical examples, as well as a greater emphasis on and an awareness of the unity and overlap between computer science and mathematics are some of the ways in which the teaching and learning of mathematics can be enhanced. Looking at the objectives of a course in mathematics, it is imperative to find out where students consistently have difficulty. Indeed, since mathematics is inspired by real objects, it follows that there is necessarily in all cases a tangible representation for any given concept, and this seems intuitively obvious in Euclidean geometry (Courant et al., 1996; Antonini, 2019). From a student development perspective, the use of such manipulative objects can help to dilute the abstract coverage of mathematics to reveal its very real relationship to physical reality, thereby making the concepts cognitively more accessible to students whose understanding has not yet reached the maturity required to comprehend its abstract nature.

Research in the field of mathematical education has shown that students who experience the use of concrete material to support the understanding of mathematical concepts perform better than students in the complementary group (Moyer-Packenham et al., 2011). This teaching strategy also has the merit of supporting students with a preference for processing material visually and kinaesthetically, while ensuring the construction of bridges between mathematical concepts and real situations (Nguyen & Cortes, 2013). I concur with the view that "teaching is based on the belief that students learn best when they appropriate knowledge through exploration and active learning" (Nguyen & Cortes, 2013). This means that teaching practice should focus on encouraging students to reflect and justify their reflections instead of memorising and stating facts.

Why include mathematical programming in second-year mathematics?

A mathematical programming course is introduced in the second-year mathematics programme at Rhodes University. This mathematical programming course is required for all students enrolled for a degree in pure mathematics and applied mathematics. It aims to familiarise students with programming mathematical concepts with computers, a tool that is omnipresent in science, in companies and even in education (Misfeldt et al., 2019). Also, this mathematical programming course is a space to prepare students for third-year courses such as numerical analysis, which need programming capabilities and theoretical thinking. Also, the objective of introducing a mathematical programming course in second-year mathematics is to provide the foundations for a solid training in scientific computing. Mathematical programming at second-year mathematics level constitutes a basic training for those preparing for a career as a data scientist or AI (artificial intelligence) scientist. The course is intensive and ambitious; it aims to develop the methods, results, and principles that every mathematician must master to become a data scientist, in accordance with international standards (Beaton, 1996).

The constructive alignment problem

The Mathematical Programming course runs across the whole academic year and each semester is assessed separately. This course requires critical thinking in Applied Mathematics and Computer Science, e.g., algebra, calculus, sequences and series, differential equations, data structure and algorithms, Python programming, etc. Students taking this course come from at least four different departments (i.e., Departments of Mathematics, Physics, Statistics and Computer Science). The Mathematical Programming course is compulsory for students majoring in Pure and Applied Mathematics. The course is optional for students majoring in other subjects, e.g., Computer Science, Physics, and Statistics. Second-year students who will be majoring in Computer Science are required to take a first-year programming course offered by the Department of Computer Science; this

means that these students are knowledgeable about programming which, combined with the first-year Algebra and Calculus courses, provide Computer Science majors with the requisite skills to learn the second-year Mathematical Programming course. The first-year courses on series, sequences and differential equations are not compulsory for Computer Science majors. Therefore, we might have some Computer Science students who do not have the adequate mathematical background to take the Mathematical Programming course to an advanced level. For those majoring in subjects other than Computer Science, there is no equivalent pre-requisite programming course available to them before they can enrol in the Mathematical Programming course in their second year. Note that for this Mathematical Programming course, programming is used as a tool to implement mathematical concepts or theories. The curriculum is designed in such a way that programming is assumed to be known and is not part of the course outcomes but it is part of the teaching and the assessment methods. It is clear that there is a gap in the constructive alignment of the course: there is no alignment between teaching objectives, assessment, and activities (Biggs, 1996). This gap in the curriculum design can impede the building of cumulative knowledge. Some Computer Science students are not strong enough in mathematics while other students may not yet have been exposed to programming; yet, they must sit in the same classroom to follow the Mathematical Programming course which requires a solid mathematical and programming background.

The students are expected to have the prerequisite academic background to be confronted with challenging course content in robust mathematics and programming. The lecturer is not supposed to teach programming, neither is the lecturer supposed to teach mathematics to the students. Instead, the lecturer is required to start the course with real-life problem-solving exercises that need robust thinking skills in programming and mathematics. In practice, however, most of the students expect the lecturer to teach programming, while others expect the lecturer to teach mathematics; neither being the actual curriculum of the course as it was originally designed.

Standard teaching approach and student failure

As noted above, the goal of this course is not to teach programming or mathematics, but to teach students how to solve real-world problems using mathematics and computers. This means that students are expected to be strong enough in Mathematics and Computer Science. The teaching method which I have adopted is to provide a brief introduction to programming and a refresher on standard algebra during the first and second lectures. I emphasise to the students that the introductory lessons cover basic concepts and that they are responsible for acquiring more advanced skills in programming and algebra independently. From the third lecture, I start providing solid and challenging exercises that I solve in the classroom using mathematics and programming. Only then can I begin to address the different levels of ability and the intersections of the disciplinary knowledge in the classroom. Individual progress is tracked by providing students with a weekly problem set, and I expect each student to submit their solutions as part of their summative assessments that I use to assess their learning at the end of the year. I also compare the performance of students on problem sets to that of their peers who enter the course with different levels of disciplinary knowledge. At the end of each week, I provide the solution to the problem sets, and encourage students to consult me in case they find any challenges in understanding the solution to a problem set.

Most of the students found the materials and the approach to the course uncomfortable to follow. This is because of the poor programming and mathematics skills of students majoring in Mathematics and Computer Science, respectively. I explained to the students why my approach or the materials for the course are relevant and indispensable, encouraging them to remain with the approach and that the learning outcomes will be to their benefit. Even after motivating them, I still had a lot of complaints; mostly from students without any programming background, as shown by the online surveys. I evaluated the course at the end of the first semester. In this online survey, eight students out of thirty-eight students responded to the survey questions. We see from this data that there are three categories of students in this class:

1. The first category represents approximately 80% of students who are comfortable with mathematics but are finding the programming aspect of the course very challenging. The majority of these students felt that I expected them to have a deep knowledge of programming. For example, one said "*... We were not taught how to code but rather reminded of the maths concepts we've already done and expected to know how to code them.*" This concern, however, seemed to be largely experienced by students who had not completed prior courses in programming and therefore could not follow the content being introduced.
2. The second category shows about 18% of students who are comfortable in mathematics and programming. These students did not have any complaints, but they instead congratulated me for the good teaching practice. This is evident in the following: "*Marcel is a great lecturer, he made me to be more interested in the mathematical aspect of coding seeing that computer science gets more theoretical and I'm more of a practical student. Hoping that I get to take his machine learning course in post-grad*".
3. In the third category, about 2% are students who are finding mathematics a bit challenging but are also somewhat comfortable with the programming. This category of students required that I should provide them with more lectures to put them on the right level: "*We are okay when the theory behind the coding is explained, more time is needed for this course so that we can discuss both the programming and mathematical aspect*" or "*It's just important for our class to have programming skills hence I think the approach to mathematical programming should be similar to that of computer science. More lectures per week could help.*"

The first category of students is mostly those majoring in Mathematics, Physics, and Statistics. The second category is probably those who are majoring in Computer Science and were exposed to sequences, series, and differential equation courses in their first-year course, and the third category are students majoring in Computer Science who did not study sequences, series, and differential equation courses but took

only Algebra and Calculus during their first-year course. This analysis acknowledges a significant limitation in the survey response rate of 21.1% (8 out of 38 students). While this presents challenges for statistical reliability, including a larger margin of error and disproportionate weight of individual responses, the findings align with classroom observations and are supported by student performance data. As shown in Figures 10.4 and 10.5, the academic performance records for 2018 and 2019 demonstrate consistently low achievement patterns, with several students receiving scores of zero, attributable to non-participation in formative assessments.

Adapting to students' learning: teaching approach and student failure

During the second semester of the same year, in reaction to the difficulties outlined above, I completely changed my approach with a focus on basic computer programming with very little mathematics. My teaching goal that semester was to focus mainly the basics of programming since over 80% of the class struggled with that aspect of the course. Solving problems relied solely on understanding basic programming applied to very simple mathematical methods. My goal was to increase the programming ability of the majority of the students in class to a more acceptable level of competence before embarking on difficult sets of problems. But this method had many drawbacks when examining the intersections of disciplinary knowledge in the classroom, as we shall see below. At the end of the semester, I administered an *"End of Semester"* questionnaire about the new approach and content of the course. Out of the thirty-eight students enrolled in the course, thirteen responded to the questionnaire.

From the questionnaire responses, I noted the following:

1. A few Computer Science students did not find the course challenging. This is because most of the focus this semester was on basic programming for which these students already had a strong background. For example, one said: "*I was a computer science student and I felt that it was easier this semester compared to the last. I remember there was one particularly challenging practice, I received a 67% I believe*" and another said

"*Too easy.... I did not need much application, maybe it is because I am a computer science student as well*".

2. Most students majoring in Mathematics, Physics, and Statistics found the approach and content of the course very interesting as they were able to understand. The focus on programming equipped these students with solid programming background that was needed to fill in the gaps. This is noted in the following remarks: "*The lectures were great this term compared to last semester. The content (problem set work) was explained clearly in class and that helped. And we were given enough time to complete the problem sets.*" and "*This term I understood Python better. Last term I was all over the place and had not much of an idea what was going on in class. Everything seemed so advance especially being someone who has never programmed before*."
3. On the other hand, some of the students majoring in Mathematics, Physics, and Statistics still encountered difficulties in understanding the programming. For example, a student stated that: "*I did a little more, but unfortunately only to some extent. I really have no interest in it and no background. So even if things were simplified, I did not find it more helpful or understandable as a lecturer would want*".

Upon reflection it is evident that this approach did not work as well as I expected. The mathematical aspect of the course was neglected to focus mostly on programming. This means that students' mathematical knowledge did not meet the expected international standard for their educational level. The students majoring in Computer Science did not benefit much since they were already exposed to the programming aspect in a first-year course. This was clearly visible during lectures as most of the Computer Science students were absent from class. It was evident that I needed to develop an alternative teaching approach that benefited both students no matter their background.

In the following section, I present and analyse the impact of an alternative teaching approach that I am proposing for the course. To evaluate the approach, I collected data from a survey offered to my students, then analysed the data while correlating the results of the analysis with students' assessment results.

I also compared students' results with those of students from previous years.

The experiment

From the above description, I have recognised that there are three groups of students with different levels of knowledge and competence and who therefore experience different difficulties. These students must share the same classroom to study the second-year mathematical programming course. I have identified the gaps in skills amongst these students and have come to conclude that the students will need different focused assistance from the lecturer - at least at the beginning of the course. In this section, I discuss an alternative approach that can be used to fill the gaps in the knowledge and competencies needed by these students to sit in the same classroom and follow the mathematical programming course. The approach is a practical simulation of Vygotsky's theory of Cognitive Development and the Zone of Proximal Development (ZPD), (Bruner, 1984). In this sense, I see the ZPD as very important since it allows me to know what each group of students can achieve with or without my guidance. It is evident that the ZPD is different for each group as shown in **Table 10.1.**

Table 10.1: Example of the ZPD

	What is known	**What is unknown**
Category 1	Mathematics: Algebra, Calculus, Analysis, Series, Sequences	Basic programming and Python
Category 2	Mathematics: Algebra, Calculus, Analysis, Basic programming and Python	Robust mathematical programming with Python
Category 3	Basic programming and Python	Mathematics: Analysis,

In Table 10.1, the "What is unknown" column denotes the skills that each category of students can master with the assistance of the lecturer. The lecturer must provide an appropriate teaching and learning strategy so that each category of students has the

opportunity to achieve the same level of knowledge before the core curriculum of the course is taught. As seen in Table 10.1, Category 1 and Category 3 need a specific aim at least at the beginning of learning before they can meet those in Category 2 to take the course. Category 1 students will need special assistance in basic programming and Python. Those in Category 3 need basic knowledge in Mathematical Analysis, Series, and Sequences. The students in Category 2 are at the appropriate level of knowledge and skills to take the course. To aid the learning process of the different categories of students to move through the ZPD, I am proposing the following algorithm:

1. Split the students into three different categories and assign each category two or three knowledgeable tutors including myself.
2. Split each category into subgroups which will facilitate social interactions with the tutors. Prepare supportive materials which includes theoretical and practical problem sets and tutorials. This will enable and enhance the problem-solving skills of the students as they move through the ZPD.
3. Merge the different categories when I am convinced that the students have integrated the same level of knowledge across the two different fields and can now work together.

One of the most challenging tasks of the teaching strategy is to identify when the merging of the students is possible. What will be the activities that will measure that the students are ready to be merged? Do these activities include having the students write many formative tests? If yes, how should these tests be designed to facilitate the development of equity amongst the different categories?

Description of the teaching and learning model

Figure 10.1 depicts a detailed schematic description of the model. The model is designed into three different levels:

1. **Level 1 or splitting:** On the first day of class, the students are split according to the degree for which they are majoring. Mathematics majors are put into Category 1, Computer Science majors into Category 2, while those majoring in any

subject other than Mathematics or Computer Science are put into Category 3. Categories 1 and 3 are of main concern while Category 2 as stated above have the required knowledge to take the course - students in this category do not have any problems with the course. To balance the number of students in Categories 1 and 3, students from Category 2 are placed in the Category 1 and Category 3 groups based on their own assessment of which group would enable them to gain the most benefit from the course (the red dotted line in Figure 10.1 indicates the sharing). This categorisation enables us to work with two groups of students. Note that the idea of moving Category 2 students into Categories 1 and 3 is to facilitate social interaction in the learning process (level 2) given that the tutors and the lecturer may not have enough time to interact with each student. Also, the sharing might stimulate the learning process as this will allow knowledgeable students in each category to help the less-knowledgeable students. Level L1 in Figure 10.1 presents a schematic view of the splitting and sharing that I am proposing. Figure 10.2 shows the initial and terminal phases of this level; in the beginning, students in Categories 1 and 2 shared some common mathematics knowledge. On the other hand, those in Categories 2 and 3 shared common knowledge in the field of programming, while those in Category 3 did not share any common knowledge with those in Category 1. At the end of this level, we see that we have two categories instead of three, and the two categories share a common piece of knowledge (Mathematics and Programming). Making sure that each category shares a common set of knowledge is fundamental to the learning process as stated above and will also allow me to further assess the level of learned skill sets while ensuring that equity is maintained.

2. **Level 2 or working level:** This is the main level where teaching and learning happen. This level (L2, see Figure 10.1) consists of helping each category of students to engage in the ZPD by teaching and encouraging students to solve problems relevant to each category. At this level students also engage in group work. Each group is assigned a knowledgeable tutor who helps the students in problem-solving and tutorials.

The interaction at this level spans 8 or 10 weeks and must be concluded even if the formative test scores from students are not satisfactory. Note that this course takes place throughout the year, so a maximum of 10 weeks is reasonable to start the course content in order to meet the timeframe necessary to cover the curriculum of the course. Figure 10.3 shows the various stages at this level. The initial stage in Figure 10.3 marks the start of this level where the two categories of students share a degree of common knowledge. After teaching and learning that includes problem-solving and tutorials, we make a judgement about whether the performance of the students is at an acceptable level. We then remove some students from one category and place them with students in another category (the dotted blue lines in Figure 10.1 indicate the distribution). This distribution shows that these students can share the same classroom with their peers without difficulty in Mathematics and/or Programming. We evaluate this from feedback and interaction during teaching. If the new distribution of students goes well without any student complaints, then we assess the students based on their new categories. This assessment is carried out in the form of multiple formative tests (see Figure 10.1). If our model is appropriate, then the tests will show that the amount of shareable knowledge amongst the two categories has become very large (see Figure 10.3).

3. **Level 3 or merging:** We see from Figure 10.3 that after some number of tests, the overlapping zone (sharable knowledge) from the two categories becomes wider and at some point, the two categories become a unique category. This is the merging point where these students have integrated the same level of knowledge across the two different fields and are now prepared to follow the mathematical programming course in the same lecture room.

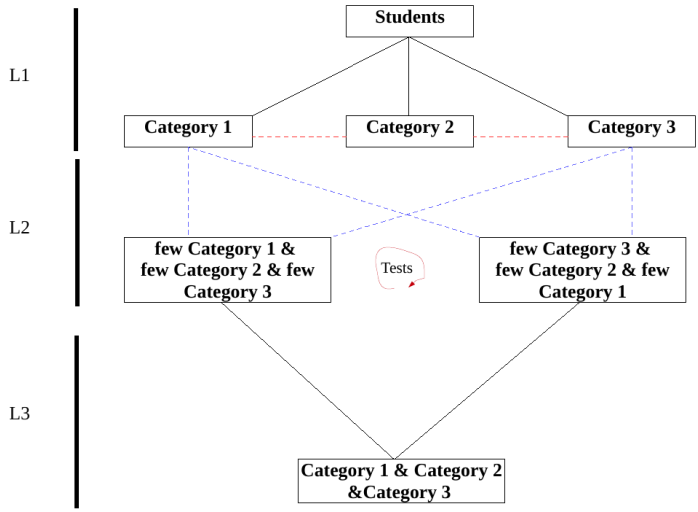

Figure 10.1: Schematics description of the learning model

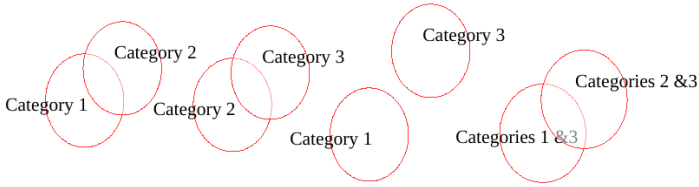

Figure 10.2: From the beginning to the end of L1 as shown in Figure 10.1

Figure 10.3: From the beginning to the end of L2 as shown in Figure 10.1.

Results

Of the 63 students who took the course during the curriculum experiment, Table 10.2 shows the number of students in each of the categories after splitting the students.

Table 10.2: Number of students per category after splitting

	Number of students
Category 1	34
Category 2	24
Category 3	5

We note in Table 10.2 that 24 students out of 63 students have the right level of proficiency to take the mathematical programming course. About thirty-four students are beginners in programming and five students do not have a good background in mathematics to take the course. As we were only two lecturers to teach the course, we had to divide the students into only two categories. We asked Category 2 students to either choose Category 1 or Category 3. Of the twenty-four students in Category 2, seven students joined Category 1 and seventeen students joined Category 3. The total number of students in Categories 1 and 2 becomes 41 and the total number of students in Categories 3 and 2 becomes 22 as shown in *Table 10.3*.

Table 10.3: Number of students per category merging the three categories into two categories.

	Number of students
Category 1 & 3	41
Category 3 & 2	22

Since the course runs throughout the academic year, we kept the split in the first semester where students in Categories 1 and 3 learned Python programming from moderate to more advanced levels, while those in Categories 3 and 2 learned solid mathematics concepts associated with programming. The problem sets were different for each category. At the end of this semester, I

administered an online "End of Semester" survey about the new approach, the course content and the different problem sets. It is noted that out of the sixty-three students enrolled in the course, fourteen responded to the survey. Out of the fourteen students who responded to the survey, ten from Categories 1 and 3, and four from Categories 3 and 2. From the questionnaire responses, I noted the following responses to the survey question:

Most students in Category 1 found the splitting, and content of the course very interesting as they were able to understand. The focus on programming equipped these students with a solid programming background that was needed to fill in the gaps. Two students from Category 1 said: "Please don't stop splitting the groups, those work wonders and introduce students to programming at the right pace.." and "math programming should made a necessity for mam 2 thus math students should do programming first year."

1. The one student in Category 3 said: "Yes, Python was easy but the concepts were difficult."
2. On the other hand, some of the students in Categories 3 and 2 said: "No, this time the maths was easier than the coding" and "No comment, good work Dr. Marcel"

The responses to the survey show that the splitting method is useful in enabling students with different levels of preparedness for the course, to meet its outcomes. Most students were satisfied with the approach. The students' end-of-semester grades in Figure 10.4 confirms the importance of the approach. The 2022 grades (red-curve) are compared to that of 2018 (black-curve) and 2019 (blue-curve). We see an increase in student performance in 2022. The model followed in the curriculum experiment appears to have eliminated the group of students who simply refused to attend lectures and were assigned zero. This is noted from flat zero blue lines that are no longer flat in 2022.

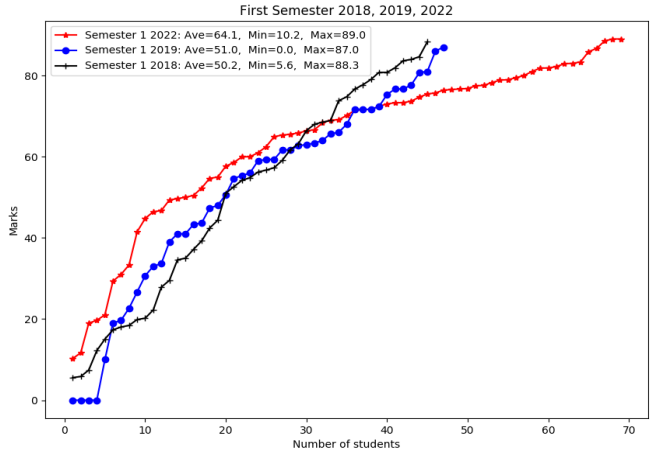

Figure 10.4: First semester mathematical programming grade for 2018, 2019, 2022

The class was merged in the second semester of 2022. We had no complaints from our students and Figure 10.5 shows their class grades at the end of the second semester compared to the second semester of 2018 and 2019. I noticed a huge improvement.

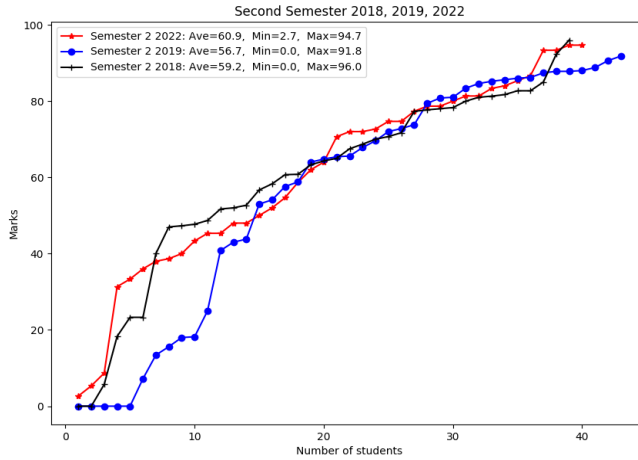

Figure 10.5: Second semester mathematical programming grade for 2018, 2019, 2022

Conclusion

Teaching methods used to teach computational mathematics should allow the greatest number of students to be able to acquire skills, curiosity, and the ability to be able to improve subsequently in each domain. I presented and discussed a teaching-learning model for a mathematical programming course introduced in the second-year mathematics curriculum. The experiment is designed to facilitate the effective movement of students within and beyond their initial ZPD. I implemented the model and provided quantitative feedback of its effectiveness. The results show that the method is effective but obliges the Department to add an additional one-hour lecture time to the course.

References

Antonini, S. 2019. Intuitive acceptance of proof by contradiction. *ZDM*, 51(5):793-806. https://doi.org/10.1007/s11858-019-01066-4

Beaton, A.E. 1996. *Mathematics Achievement in the Middle School Years. IEA's Third International Mathematics and Science Study (TIMSS).* Boston College, Center for the Study of Testing, Evaluation, and Educational Policy. Campion Hall 323, Chestnut Hill. MA 02167.

Biggs, J. 1996. Enhancing teaching through constructive alignment. *Higher Education*, 32(3):347-364. https://doi.org/10.1007/BF00138871

Bruner, J. 1984. Vygotsky's zone of proximal development: The hidden agenda. *New Directions for Child Development*, 23(1984):93–97. https://doi.org/10.1002/cd.23219842309

Courant, R., Robbins, H. & Stewart, I. 1996. *What is Mathematics?: An elementary approach to ideas and methods*. New York: Oxford University Press USA. https://doi.org/10.1093/oso/9780195105193.001.0001

Dubinsky, E. & McDonald, M.A. 2002. APOS: A constructivist theory of learning in undergraduate mathematics education research. In: Holton, D., Artigue, M., Kirchgräber, U., Hillel, J., Niss, M. & Schoenfeld, A. (eds.) *The teaching and learning of mathematics at university level*. New ICMI Study Series, vol 7. Dordrecht: Springer, pp: 275-282. https://doi.org/10.1007/0-306-47231-7_25

Hassard, J. & Dias, M. 2013. *The art of teaching science: Inquiry and innovation in middle school and high school*. (2nd ed.) New York: Routledge. https://doi.org/10.4324/9780203892961

Meyer, J.H.F. & Land, R. 2005. Threshold concepts and troublesome knowledge (2): Epistemological considerations and a conceptual framework for teaching and learning. *Higher Education*, 49(3):373-388. https://doi.org/10.1007/s10734-004-6779-5

Middendorf, J. & Pace, D. 2004. Decoding the disciplines: A model for helping students learn disciplinary ways of thinking. *New Directions for Teaching and Learning*, 98(2004):1-12. https://doi.org/10.1002/tl.142

Misfeldt, M., Szabo, A. & Helenius, O. 2019. *Surveying teachers' conception of programming as a mathematics topic following the implementation of a new mathematics curriculum.* Proceedings of the Eleventh Congress of the European Society for Research in Mathematics Education (No. 13). Freudenthal Group, Freudenthal Institute. ERME.

Moyer-Packenham, P.S., Bolyard, J.J., Oh, H. & Cerar, N.I. 2011. Common features of professional development activities for mathematics and science teachers. *Professional Development in Education*, 37(4):571-89. https://doi.org/10.1080/19415257.2010.531597

Niss, M. 2003. *Mathematical competencies and the learning of mathematics: The Danish KOM project*. Proceedings of the 3rd Mediterranean conference on mathematical education, 115-124.

Nguyen, H.T. & Cortes, M. 2013. Focus on middle school: Teaching mathematics to ELLs: Practical research-based methods and strategies. *Childhood Education*, 89(6):392-5. https://doi.org/10.1080/00094056.2013.854130

Presmeg, N. 2002. Beliefs about the nature of mathematics in the bridging of everyday and school mathematical practices. In: Leder, G.C., Pehkonen, E. & Törner, G. (eds.) Beliefs: A hidden variable in mathematics education? Mathematics Education Library, 31. Dordrecht: Springer, pp: 293-312. https://doi.org/10.1007/0-306-47958-3_17

Stylianides, G.J. & Stylianides, A.J. 2008. Proof in school mathematics: Insights from psychological research into students' ability for deductive reasoning. *Mathematical Thinking and Learning*, 10(2):103-33. https://doi.org/10.1080/10986060701854425

Toohey, S. 1999. *Designing courses for higher education*. Maidenhead, UK: McGraw-Hill Education.

Young, M. 2014. What is a curriculum and what can it do? *Curriculum Journal*, 25(1):7-13. https://doi.org/10.1080/09585176.2014.902526