How to (and why) use Python to teach introductory Calculus

Jan-Fredrik Olsen

***Abstract*—In 2015, a course in numerical computations using Python (NUMA01) was made obligatory in the first semester of math studies for all students in the Bachelor program in Mathematics at the Faculty of Science in Lund. Here, we discuss how this has influenced the course in introductory Calculus (MATA21).**

***Index Terms*—Calculus, education, programming, Python**

I. Introduction

TRADITIONALLY taught, a Calculus course usually consists of:

1. theory and proofs,
2. computational exercises.

The "theory" part of the course is the domain of the lectures where the professor proves why the formulas of Calculus are true. The students are usually left with solving some "standard" computational exercises using these formulas.

For the average student, it is hard to keep track of how the theory and computational exercises are connected. Moreover, since only "standard exercises" are likely to appear on the final exam, a one-sided focus on procedural skills is encouraged. This disconnect between theory and exercises makes it difficult for students to understand why they are supposed to solve the exercises given to them. Indeed, studies have shown that students do not show any measurable improvement in their understanding of the basic concepts of Calculus after completing a traditional first semester Calculus course, [2].

Moreover, the relevance of Calculus is undermined by the apparent limited scope of the techniques involved (e.g., the need to disregard friction to make a differential equation tractable).

It is therefore only reasonable for students to ask the question: "Why do we have to learn Calculus?"

We argue that by introducing a non-math specific programming language already in the first semester, we can use numerical computations to bridge the gap between theory and practical computations. Moreover, the numerical perspective is ideal for showcasing the relevance of Calculus in the sciences.

II. Background

Since the advent of the personal computer, there have been several pushes to make numerical computations part of the syllabus for introductory Calculus courses at the university level. Indeed, most modern Calculus textbooks include the use of Maple, Matlab or Mathematica to some extent (see, e.g, [1]).

In Sweden, it has recently become necessary to review the role of numerical computations at the university level as programming has become a part of the curricula of all science subjects at Swedish elementary and high schools since 2018, [5].

Already in 2015, the course Scientific Computations using Python (NUMA01), was made obligatory in the first semester of math studies of all students at the Bachelor programs in Mathematics and Physics at the Faculty of Science in Lund. The objective was to provide students and teachers with a tool to more efficiently perform numerical computations.

Initially, this effort was inspired by similar developments dating back to the early 2000s at the University in Oslo (UiO) where numerical computations using Python were integrated into the introductory math courses. In fact, what started as a limited reform to modernize math and physics curricula at UiO, was later elevated to a national Centre for Excellence in Education (thus receiving 240 million NOK over a 10-year period) which has now expanded to also include the study programs in Chemistry, Geology and Biology at UiO, see, e.g., [4].

III. The basic pedagogical idea

In the infancy of Calculus, the availability of computing power was highly limited. It could be argued that Newton and Leibniz developed the theory of differential and integral Calculus to compensate for this.

At its core, most of the fundamental ideas of Calculus are quite naïve and straight-forward. Indeed:

(i) Infinite series are approximated by partial sums.
(ii) Derivatives are approximated by slopes of secant lines.
(iii) The definite integral is approximated by finite sums of rectangular areas.
(iv) Using Euler's method, the solution to a differential equation is approximated by a sequence of straight lines.

The standard objective of most Calculus courses is to study how the theory of integration and differentiation emerges as the "limit" of such approximations.

The basic pedagogical idea is that since limits are probably the most difficult concepts for students to understand, why not put more emphasis on the more student friendly notions of approximation in order to motivate students?

In fact, due to the current massive availability of

Manuscript received November 1, 2018.

Jan-Fredrik Olsen is with the Centre for Mathematical Sciences, Faculty of Science, at Lund University, Box 118, 22100 Lund, Sweden (e-mail: jan-fredrik.olsen@math.lu.se).

computational power, the classically dominant techniques of differentiation and integration have become less relevant, and instead, the above list of naïve ideas has evolved into a battery of powerful numerical algorithms that are directly applicable to problems in both research and in the industry.

We also note that since the "boundary conditions" set by later courses have not changed, placing additional emphasis on approximations does lead to a danger of overloading the course.

For an overview of the history of the use of approximations as a unifying theme in introductory Calculus, see [6] and the references therein.

IV. Why Python and not Maple?

To effectively perform numerical computations, one has to use some type of programming tool. Now, there is nothing sacred about using Python. However, in our experience, the language one choses ought to satisfy the criteria:

1. Non-math specific
2. Simple (and transparent) syntax
3. Powerful math libraries
4. Open source
5. Widely used in the industry

We now address the two first criteria.

Why a non-math specific language? The fundamental reason is to control expectations. Indeed, students tend to use Maple as a powerful scientific calculator, [3]. That is, when faced with a mathematical problem, they search for a ready-made blackbox solution. In contrast, when using a non-math specific programming language with only the knowledge of a handful of commands (for, while, if, else), students are instead forced to think about the nature of whatever mathematical creature that they are facing. In short, Python provides the students with a technological "sandbox" where they are encouraged to construct their own solutions.

To illustrate that Python has a relatively simple syntax, we include some examples how some of the basic concepts of Calculus can be implemented and/or studied in Python:

- The infinite series $1+\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\cdots$ can be approximated by using the code

```
mySequence = [1/2**n for in range(0,10000)]
mySum = sum(mySequence)
```

- The formal definition of the limit $\lim_{n\to\infty} a_n = L$ can be studied using the code

```
epsilon = 1/10
n = 0
while 1/2**n >= epsilon:
    n = n + 1
```

- The definite integral $\int_0^2 x^2 dx$ can be approximated by using the code

```
X = [n/50 for n in range(0,101)
S = 0
for n in range(1,101):
    S = S + x[n]**2*(x[n]-x[n-1])
```

V. How does this idea work in practice?

A. Teacher's perspective

There is a limit to how much we could modify the course MATA21 since the subsequent math courses have not been changed. With this in mind, we initially tried to "seamlessly" integrate Python into MATA21 by interspersing numerical Python exercises among the "regular" exercises. This more or less failed since the students chose the path of least resistance, and essentially avoided the Python exercises.

As a response to this, we ended up creating a sequence of obligatory "mini-projects" where the students are to study various notions of approximations using a combination of numerical computations, using Python, and theoretical arguments (e.g., how many terms do we have to include in a partial sum for it to approximate an infinite series).

These mini-projects are placed at "transitional points" during the course and have had the effect that the course now alternates between weeks where definitions relying on approximations are explored in project-oriented work, and more classical weeks where the theory arising "in the limit" is worked out.

B. Student's perspective

Course evaluations show that students are, in general, very happy with how the course currently works. For instance, between 80 and 90% of the respondents to the course evaluation in the spring of 2018 "agree somewhat" or "agree completely" with the following statements (the other two options on the survey were "disagree somewhat" and "disagree completely"): "The course has stimulated my interest in mathematics", "The course has increased my ability to use computer based tools to understand mathematical concepts", "My prior knowledge has been sufficient to learn contents of the course", "As a result of this course, I feel confident about tackling unfamiliar problems" and "The course had a reasonable workload".

We note that the latter response was surprising, due to the fact that the course is now rather overloaded.

VI. Discussion

By placing a course in Scientific Computations using Python in the first semester, we have been able to put more emphasis on approximations in the course MATA21. Students are positive to this change, as shown by responses to course evaluations, this despite an increased workload. While the instructor feels that the level of classroom discussions has increased, we have not yet studied the impact on student learning.

References

[1] R.A. Adams and C. Essex, *Calculus: A Complete Course*, 9th ed. Prentice-Hall (Canada), 2017.

[2] J. Epstein, "The Calculus concept inventory: Measurement of the effect of teaching methodology in mathematics", *Notices of the AMS*, vol. 60, no. 8, pp. 1018-1026, 2013.

[3] MapleSoft (2018), "The Top 10 Reasons Students Use Maple". Available: https://www.maplesoft.com/products/maple/students/top10.aspx.

[4] L. Nederbragt (2017), "Experiences with the first edition of Introduction to Computational Modelling for the Biosciences". Available: https://flxlexblog.wordpress.com.

[5] Skolverket (2018), "Förändringar och digital kompetens i styrdokument". Available: https://www.skolverket.se/temasidor/digitalisering/digital-kompetens.

[6] K.S. Sofronas, et al, "A study of Calculus instructors' perceptions of approximation as a unifying thread of the first-year Calculus", *Int. J. Res. Undergrad Math. Ed.*, vol. 1, pp. 386-412, 2015.