Athens Journal of Mediterranean Studies 2027, 13: 1-12
https://doi.org/10.30958/ajms.X-Y-Z

Approximate Integration in Java and Python

*By Mauro Pullin**

This article describes a teaching-oriented activity for first-year students in Electronic Engineering and Computer Engineering enrolled in Fundamentals of Computer Science 1 at the University of Padua. The activity uses numerical integration to reinforce students' understanding of iterative loops and to revisit concepts related to definite and indefinite integrals introduced in Mathematical Analysis 1. Three numerical methods — the rectangle rule, the trapezoidal rule, and Simpson's rule — are discussed, with implementations in Java and Python. By translating mathematical procedures into algorithms, students practise loop construction, numerical reasoning, and the interpretation of approximation error. The article also considers the potential educational value of experiential programming activities, while recognizing that claims about student outcomes require systematic evidence (Quapp & Holschemacher 2022).

Keywords: *numerical integration, rectangle rule, trapezoidal rule, Simpson's rule, Java, Python, programming education*

Introduction

The purpose of this work is primarily educational and refers to some activities conducted with first-year university students in the degree programs in Electronic Engineering and Computer Engineering, majoring in "Fundamentals of Computer Science" (I taught this subject at the University of Padua for five years). This is a practice-oriented educational project.

In Italy, first-year university students attend various high schools, where the fundamentals of computer science, and particularly programming, are not always taught. In the course "Fundamentals of Computer Science 1", the learning objectives of the teaching unit described in this article consist of an in-depth understanding of the programming language and the study of the notions of iterative loops and approximate calculus. Furthermore, the notion of indefinite integrals, rigorously studied in the course "Mathematical Analysis 1", which is taken in the first year of engineering, is analyzed and reviewed.

Prior Knowledge and Lesson Plan

A possible lesson plan could include a theoretical introduction to approximate integration methodologies, also presented through the use of graphs, pseudo-coding of algorithms using a structured language or flowcharts, and subsequent coding in the Java and Python programming languages.

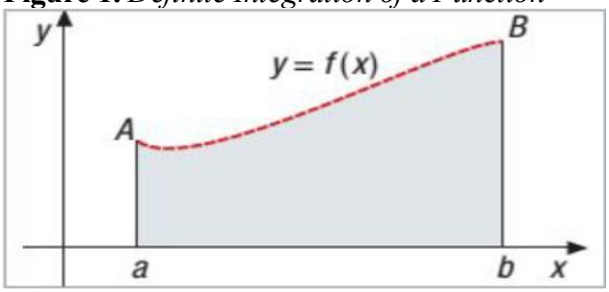
*Adjunct Professor, University of Padova, Italy.

The required prior mathematical knowledge includes the definitions of functions, definite integrals, indefinite integrals, and the fundamental theorem of integration. The required prior computer science knowledge includes the concept of algorithms, the representation of algorithms using pseudo-coding or flowcharts, introductory notions of programming languages, the concepts of integer and floating-point variables, conditional structure, and iterative loops.

Students taking a computer science course on programming languages in their first year of college learn some important concepts, including conditional structures and iterative loops (Savitch 1997, Downey 2002–2012). An iterative loop is composed of a set of instructions that must be executed multiple times to complete an operation. It is very instructive to analyze the functioning of iterative loops by solving real-world problems, such as those originating in mathematics. In general, an iterative loop can be of three types: a "for" loop, which is a loop with a counter; a "while" loop, which is a loop with a condition at the beginning; and a "do...while" loop, which is a loop with a condition at the end (Savitch 1997).

The rigorous definitions of indefinite integrals and definite integrals are taught in the Mathematical Analysis 1 course, which is taken in the first year of engineering. These definitions are not reported in this work because they are already well known to students. A non-rigorous definition of a definite integral, but useful for the purposes of this work, reminds us that the definite integral between the endpoints a and b of the function y=f(x) represents the signed area of a trapezoid (Figure 1) (Mbobi 2012).

Figure 1. *Definite Integration of a Function*



The fundamental theorem of integration allows, under appropriate assumptions (studied in the Mathematical Analysis 1 course), to calculate the value of the definite integral starting from the primitive F(x) of the function f(x), which is determined through indefinite integration. We can write (Bergamini & Barozzi 2024):

$$\int_a^b f(x)\,dx = F(b) - F(a)$$

where $F(x) = \int f(x)\,dx$

However, an antiderivative cannot always be expressed in elementary functions. For example, the integrals below have antiderivatives, but these antiderivatives are not expressible in elementary functions:

$$\int e^{-x^2}\,dx = ?$$

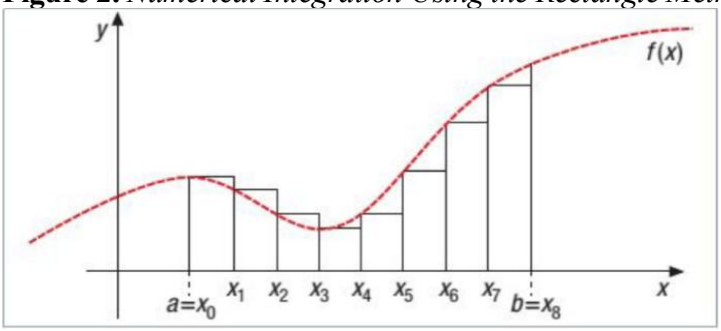
$\int (\sin x) / x \, dx = ?$

In these cases it is necessary to resort to numerical integration (Camagni & Nikolassy 2023).

The Rectangle Rule

Teaching experience has shown that the study and implementation of numerical integration methods promote both computer science learning and a better understanding of mathematical concepts (Bergamini & Barozzi 2024). In the rectangle rule, the region to be integrated is divided into n rectangular parts, all having the same width h = (b–a)/n, where h is called the integration step. Figure 2 graphically represents the rectangle rule.

Figure 2. *Numerical Integration Using the Rectangle Method*



The sum of the rectangle areas approximates the signed integral. In general, increasing the number n of subintervals improves the approximation for sufficiently regular functions, although the error depends on the function and the chosen rule.

Computer Implementation in Java Language

We know that teaching programming is not just about preparing future computer science experts; it is also about teaching students how to reason. Programming means breaking down a complex problem into simpler parts, identifying relationships, and building solutions step by step. In other words, it means developing logical thinking, analytical skills, and a critical mind.

For the computer science implementation of the algorithm analyzed, the Java and Python programming languages were chosen. Below, we see the Java source code for the integration of Archimedes' parabola, which is said to be exactly integrable (recall that its definite integral calculated between -1 and 1 is 4/3).

```
import java.util.Scanner;
import java.math.*;
public class IntegrazioneRettangolare {
   public static void main(String args[]) {
      Scanner tastiera = new Scanner(System.in);
      double h, x, y, AreaTot = 0.0, AreaRett = 0.0;
      int a = -1.0, b = 1.0, N, k, x, y;
      do {
         System.out.print("Insert N: ");
         N = tastiera.nextDouble();
         if (N <= 0) {
            System.out.println("N is wrong, it must be positive");
         }
      } while(N <= 0);
      System.out.print("Insert the lower bound a: ");
      a = tastiera.nextDouble();
      do {
         System.out.print("Insert the upper bound b: ");
         b = tastiera.nextDouble();
         if (b < a) {
            System.out.println("Incorrect upper bound");
         }
      } while(b < a);
      h = (b - a) / N;
      for(k = 0; k < N; k++) {
         x = a + k * h;
         y = AltezzaParabola(x);
         AreaRett = h * y;
         AreaTot = AreaTot + AreaRett;
      }
      System.out.println("Result = " + AreaTot);
   }
   public static double AltezzaParabola(double x) {
      double y;
      y = 1 - Math.pow(x,2);
      return y;
   }
}
```

Computer Implementation in Python Language

In recent years, Python has established itself as one of the most effective teaching tools (Downey 2002-2012). And this is no coincidence. Python is a language designed to be simple, readable, and close to human language. Its code is clean, essential, and free of superfluous elements. This allows students to focus on what really matters: reasoning.

From an educational perspective, Python has an important advantage: it reduces cognitive load. Students do not have to waste energy understanding complex syntax but can focus on the problem at hand. This way, the focus of learning returns to reasoning. Python is therefore a tool that makes programming accessible. It lowers the entry threshold without lowering the level. It allows a greater number of students to acquire these skills, even without advanced technical training (Downey 2002-2012).

Below is the Python implementation of the rectangle rule, relating to the integration of Archimedes' parabola.

```
import math

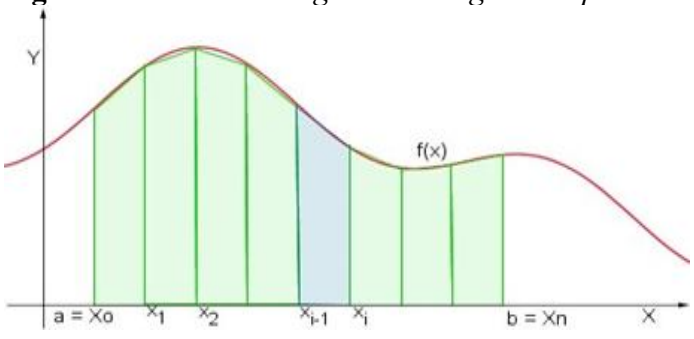
def AltezzaParabola(x):
    f = 1 - math.pow(x,2)
    return f

AreaTot = 0.0
AreaRett = 0.0
a = -1.0
b = 1.0
N = int(input("Insert N: "))
while N <= 0:
    print("N is wrong, it must be positive")
    N = int(input("Insert N: "))
a = float(input("Insert the lower bound a: "))
b = float(input("Insert the upper bound b: "))
while b < a:
    print("Incorrect upper bound")
    b = float(input("Insert the upper bound b: "))
h = (b - a) / N
for k in range(N):
    x = a + k * h
    y = AltezzaParabola(x)
    AreaRett = h * y
    AreaTot = AreaTot + AreaRett
print("Result = ", AreaTot)
```

The Trapezoidal Method

In the trapezoidal rule, the integration interval [a,b] is divided into n equal parts and the area of the right trapezoids with height h = (b–a)/n and bases given by the value of f(x) at the ends of each subinterval into which the integration interval has been divided is calculated. Figure 3 graphically represents the trapezoidal rule.

Figure 3. *Numerical Integration Using the Trapezoidal Method*



Computer Implementation in Java Language

Below is the Java source code for the trapezoidal integration of Archimedes' parabola.

```
import java.util.Scanner;
import java.math.*;
public class IntegrazioneTrapezoidale {
   public static void main(String args[]) {
      Scanner tastiera = new Scanner(System.in);
      double a = -1.0, b = 1.0, N, h, k, x, y1, y2, AreaTot = 0.0, AreaTrap;
      do {
         System.out.print("Insert N: ");
         N = tastiera.nextDouble();
         if (N <= 0) {
            System.out.println("N is wrong, it must be positive");
         }
      } while(N <= 0);
      System.out.print("Insert the lower bound a: ");
      a = tastiera.nextDouble();
      do {
         System.out.print("Insert the upper bound b: ");
         b = tastiera.nextDouble();
         if (b < a) {
            System.out.println("Incorrect upper bound");
         }
      } while(b < a);
      h = (b - a) / N;
      for(k = 0; k < N; k++) {
         x = a + k * h;
         y1 = AltezzaParabola(x);
         x = a + (k + 1) * h;
         y2 = AltezzaParabola(x);
         AreaTrap = (y1 + y2) * h / 2;
         AreaTot = AreaTot + AreaTrap;
      }
      System.out.println("Result = " + AreaTot);
   }
   public static double AltezzaParabola(double x) {
      double y;
      y = 1 - Math.pow(x,2);
      return y;
   }
}
```

Computer Implementation in Python Language

The Python programming language was also chosen for the computer implementation of the algorithm analyzed. Here is the Python implementation of the trapezoidal rule:

```
import math

def AltezzaParabola(x):
   f = 1 - math.pow(x,2)
   return f

AreaTot = 0.0
AreaTrap = 0.0
a = -1.0
b = 1.0
N = int(input("Insert N: "))
while N <= 0:
   print("N is wrong, it must be positive")
   N = int(input("Insert N: "))
a = float(input("Insert the lower bound a: "))
b = float(input("Insert the upper bound b: "))
while b < a:
   print("Incorrect upper bound")
   b = float(input("Insert the upper bound b: "))
h = (b - a) / N
for k in range(N):
   x1 = a + k * h
   y1 = AltezzaParabola(x1)
   x2 = a + (k + 1) * h
   y2 = AltezzaParabola(x2)
   AreaTrap = (y1 + y2) * h / 2
   AreaTot = AreaTot + AreaTrap
print("Result = ", AreaTot)
```

Empirical Evaluation of Errors in the Method of Rectangles and Trapezoids

Numerical calculation includes approximate calculation methods, which can therefore introduce errors. (Bergamini & Barozzi 2024, Press et al. 2007) It is intuitively understood that both the rectangle rule and the trapezoidal rule introduce errors because the surface whose area is to be calculated is not completely "covered." It is also intuitively understood that the error decreases as the number N of intervals increases. We know that the integral between the extremes –1 and 1 of Archimedes' parabola is 4/3, or 1.333… (This is a periodic number). By inserting –1 and 1 as the extremes, the program implementing the rectangle rule was activated, varying the value of the number of intervals N. The results obtained are shown in Table 1.

Table 1. *Value of the Integral and Error as a Function of the Number of Intervals*

N	Value of the integral	Error
5	1.280000	0.053333
10	1.320000	0.013333
15	1.328889	0.004444
20	1.330000	0.003333

Note: For $f(x) = 1 - x^2$ on $[-1, 1]$, the exact integral is 4/3. The composite left-rectangle and trapezoidal rules produce the same values in this particular example because $f(-1) = f(1) = 0$; this coincidence should not be generalized to arbitrary functions.

By always entering -1 and 1 as the endpoints, the program implementing the trapezoidal rule was activated, varying the value of the number of intervals N. The results obtained are shown in Table 2.

Table 2. *Value of the Integral and Error as a Function of the Number of Intervals*

N	Value of the integral	Error
5	1.280000	0.053333
10	1.320000	0.013333
15	1.328889	0.004444
20	1.330000	0.003333

It can be observed that the error decreases as N increases, furthermore it is experimentally proven that the trapezoidal rule is more precise than the rectangle rule.

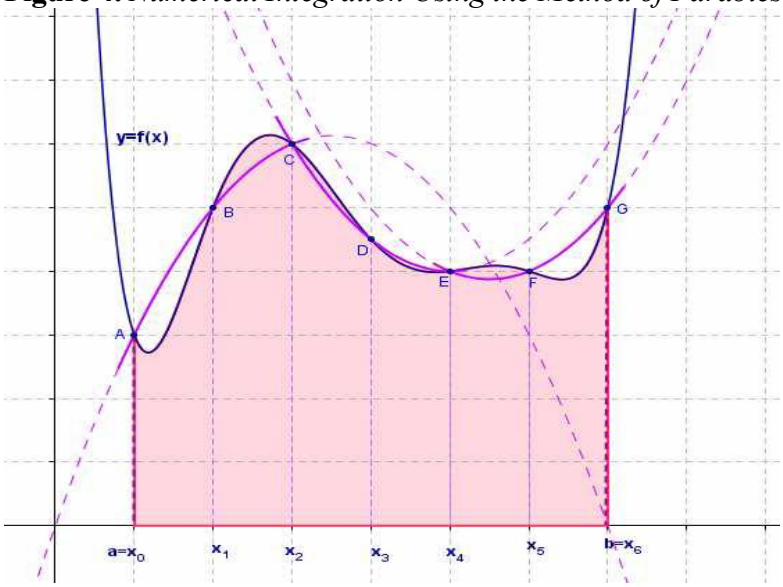
Simpson's Rule

Regarding approximate integration using the parabola method, the basic idea is simple. If a curve is complex, we can try to replace it, at least locally, with something simpler. In this case, we do not use a straight line, but a parabola. A parabola is the simplest polynomial capable of "following" the curve representing the behavior of a function well. With the coordinates of three points, we can calculate the equation of a parabola passing through them, and this parabola can approximate the function much more precisely than a simple straight line.

The method works like this: divide the interval into subintervals, take three points at a time, and construct a parabola passing through these points. Then calculate the area under the parabola, which is easy to determine, and add all the contributions. The result is an estimate of the definite integral.

From an educational perspective, the parabola method is valuable because it allows for an easy connection between geometry and analysis. Students see concretely how a curve can be approximated by simpler shapes. This is not just a calculation, but a visual representation. To this end, consider Figure 4, which illustrates the graph of a function. We wish to calculate the definite integral between the endpoints a and b of the function whose graph it represents.

Figure 4. *Numerical Integration Using the Method of Parables*



Let us suppose that we have divided the integration interval [a,b] into n equal parts (n being an even integer) and that we have calculated the ordinates corresponding to all the "nodes" x_i, which will be the coordinates of the points A(a;f(a)), B(x_1;f(x_1)), C(x_2;f(x_2)), D(x_3;f(x_3)), ...

In the rectangle and trapezoidal rules, we discussed the "construction of rectangles and trapezoids" for each of the subdivision intervals. We could also have said that, in each interval, the integrand function f(x) is replaced by a more elementary function: by the function y=k (a constant function) for the rectangle rule and by the function y=mx+p (a linear function) for the trapezoidal rule.

In Simpson's rule, the integrand is approximated over each pair of consecutive subintervals by a quadratic polynomial interpolating the function at three equally spaced nodes.

The composite Simpson's rule approximates the definite integral over [a,b] using an even number n of equal subintervals (Bergamini & Barozzi, 2024):

$$Area = \frac{\Delta x}{3}\left[f(a) + f(b) + 2 \cdot \sum_{i=1}^{k-1} f(x_{i\ pari}) + 4 \cdot \sum_{i=1}^{k} f(x_{i\ dispari}) \right]$$

N.B.: For ease of writing, let k=n/2 (n even; pari = even, dispari = odd)

Computer Implementation

As previously noted, along with the rectangle and trapezoidal rules, the parabola method constitutes the set of numerical integration methods known as Newton-Cotes methods, whose main characteristic is the division of the integration interval [a,b] into

n equal parts. While in the other two methods, the number n of subdivisions can be any integer, for Simpson's method, the number n of subdivisions must be even.

For educational reasons, it was decided to leave the implementation of the parabola method with the Java and Python programming languages to the students. Students were provided only with the initial part of the program in Java and a function written in Python. Below is what was provided to the students.

```
// JAVA
// acquisition and control cycle of a, b
do {
   System.out.print("Insert the first extreme of integration: ");
   a = tastiera.nextDouble();
   System.out.print("Insert the second extreme of integration: ");
   b = tastiera.nextDouble();
   if (b <= a) {
      System.out.println("Incorrect integration extremes ");
   }
} while(b <= a);

// acquisition and control cycle of n
do {
   // acquisizione n
   System.out.print("Enter n as positive and even: ");
   n = tastiera.nextInt();
   if (n <= 0 || n % 2 != 0) {
      System.out.println("Incorrect value of n ")
   }
} while(n <= 0 || n % 2 != 0);

// calculating the amplitude h of an interval
h = (double) (b - a) / n;

// to be completed by students...
…

# PYTHON
def simpson(f, a, b, n):
   if n % 2 != 0:
      raise ValueError("n deve essere pari")

   dx = (b - a) / n
   area = f(a) + f(b)

   for i in range(1, n):
      x = a + i * dx
      if i % 2 == 0:
         area += 2 * f(x)
      else:
         area += 4 * f(x)

   return area * dx / 3
```

Educational Insights

Below, we would like to explore some important educational aspects related to this work. First, it is important to help students understand the geometric meaning of the definite integral, such as the signed area subtended by a curve representing a function. Furthermore, it is essential to understand why, in many cases, the integral cannot be solved analytically and numerical methods are used (Press et al. 2007). Other works that address the pedagogical aspects of numerical integration include those of Morigi (2024) and Webb (2024).

Three methods for approximating the integral were examined and compared: rectangles, trapezoids, and parabolas. The approach described in this work fosters the connection between visual and numerical reasoning, developing the relationship between abstract concepts and practical calculation. The educational activities described in this work have proven very useful for exploring both the mathematical and computer science aspects.

Numerical calculation is a branch of mathematics and computer science that deals with finding approximate solutions to complex problems, using algorithmic procedures and computational tools implemented on computers. It could be said that numerical computation is the bridge between theoretical mathematics and the real world. A key aspect to consider is that numerical computation goes beyond simply "doing calculations"; it introduces a new way of thinking: algorithmic thinking.

This approach is the basis of how modern computers work. It is no coincidence that numerical computation is closely linked to computer science. Without numerical algorithms, there would be no scientific simulations, 3D graphics, artificial intelligence, weather forecasting, or advanced engineering models. We could say that much of the technology we use today is based on numerical methods.

Another key concept of numerical computation is approximation (Press et al. 2007). Unlike pure mathematics, where the goal is absolute accuracy, in numerical computation, the idea is accepted that every result has a margin of error. But this error is not a problem: it is a quantity that can be studied, controlled, and reduced (Morigi 2024).

From an educational perspective, numerical computation has enormous value. First and foremost, it helps connect mathematics to reality. Students no longer work solely with abstract symbols, but see concrete applications: data, simulations, models. Second, it develops computational thinking. Students learn to build algorithms, think step by step, and verify results. Furthermore, it introduces a very important concept: mathematics is not always exact, but it is always useful. This profoundly changes the perception of the discipline.

Another fundamental aspect is the role of digital tools. Today, numerical computation is closely linked to computers and programming languages. Without tools like Java, Python, or other programming languages, many numerical methods would be too long or complex to apply manually. But it is important to emphasize that computer science and computers do not replace mathematical thinking: they amplify it. The role of the student or researcher remains central in developing the method and interpreting the results.

Conclusion

In conclusion, we can say that numerical computation is one of the most powerful tools of modern applied mathematics. Not because it renounces precision, but because it embraces the complexity of the real world and addresses it with tools that are at once flexible, efficient, and rigorous.

In a certain sense, numerical calculus teaches us an important lesson: it is not always possible to obtain a perfect answer, but it is always possible to approach the truth with ever-increasing precision. And this, in science as in life, makes a big difference.

The implementation of the methods for approximate integration, described in this work, also achieves an important educational result: it stimulates conscious, experiential, and intrapersonal development, open-mindedness, and reflection, contributing to the improvement of students' self-esteem (Quapp & Holschemacher 2022).

Teaching numerical calculus is not just about teaching mathematics and computer science, but about offering students a concrete growth experience. An experience in which they learn not only to solve problems, but also to believe they can do so (Quapp & Holschemacher 2022, Press et al. 2007).

References

Bergamini M, Barozzi G (2024) *Matematica.blu 2.0 2^ Ed. – Volume 5*. Zanichelli Editore.

Camagni P, Nikolassy R (2023) *InfoM@T per il Liceo Scientifico Opzione Scienze Applicate*. (InfoM@T for the *Liceo Scientifico* (Scientific High School) – Applied Sciences Option). Hoepli Editore.

Downey AB (2002–2012) *Think Python: How to Think Like a Computer Scientist*. O'Reilly Media.

Mbobi A (2012) *Principles and Applications of Numerical Integration Methods*. LAP LAMBERT Academic Publishing.

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007) *Numerical Recipes: The Art of Scientific Computing*. 3rd Edition. Cambridge University Press.

Quapp U, Holschemacher K (2022) Management Strategies for Engineering Faculties under Consideration of Current Developments in the Higher Education Sector. *Athens Journal of Technology and Engineering* 9(2): 147–158.

Savitch W (1997) *Java: An Introduction to Problem Solving and Programming*. Pearson Education.

Morigi S (2024) *Numerical Differentiation and Integration*. University of Bologna "Alma Mater Studiorum".

Webb K (2024) *Numerical Methods for Engineers*. Oregon State University-Cascades.