Approximate Integration Defined in Java and Python – An Educational Approach

The aim of this work is mainly educational and refers to some activities carried out with students attending the first year of university. This work describes how to calculate the definite integral of a function, which cannot be integrated analytically, using the rectangle method and the trapezoidal method. The computer implementation of the analyzed algorithms using the Java and Python programming languages is then discussed.

Keywords: *definite integral, indefinite integral, numerical integration, rectangular integration, rectangle method, trapezoidal integration, trapezoid method, method of parabolas.*

Introduction

In Italy, first-year university students attended a variety of high schools, where they were not always taught the fundamentals of computer science, and programming in particular. Students who take a computer science course on programming languages in their first year of university learn about some important concepts, among which are conditional structures and iterative loops.

An iterative loop is composed of a set of instructions that must be executed multiple times to complete an operation.

It's very instructive to analyze how iterative loops work by solving real-world problems, such as those originating in mathematics.

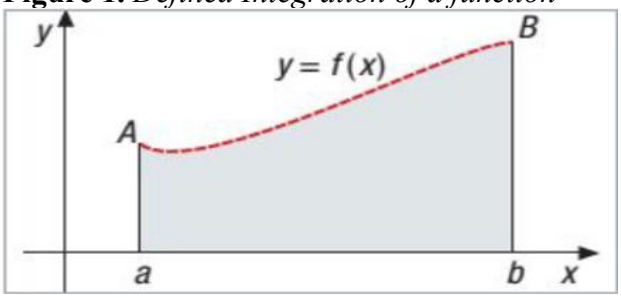
In general, an iterative loop can be of three types:

- "for" loop, or a loop with a counter;
- "while" loop, or a loop with a conditional at the beginning;
- "do...while" loop, or a loop with a conditional at the end.

The approximate calculation of definite integrals is one of the problems that can be solved with the help of computer science and, in particular, with the use of iterative loops.

We recall that the definite integral between a and b of the function y=f(x) represents the area (with sign) of a trapezoid (figure 1).

Figure 1. *Defined Integration of a function*



The fundamental theorem of integral calculus allows, under appropriate assumptions, to calculate the value of the definite integral from the antiderivative F(x) of the function f(x), which is determined through indefinite integration.

We can write:

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

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

It may happen, however, that the primitive F(x) cannot be determined using analytical methods.

For example, the following indefinite integrals cannot be calculated:

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

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

In these cases, it is necessary to resort to numerical integration.

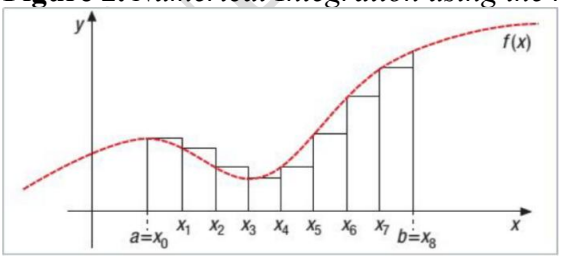
The method of rectangles

Introduction

Teaching experience has shown that studying and implementing numerical integration methods promotes both the learning of computer science and a better understanding of mathematical concepts.

In the rectangle method, 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. *Numerical Integration using the rectangle method*



Unless there is an error, the sum of the areas of the rectangles equals the area under the curve with the equation y=f(x). It is intuitively understood that the error decreases as the number n of intervals increases.

Computer implementation in Java language

We know we live in an increasingly digital world, but teaching programming isn't just about preparing future computer science experts. It's, above all, 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.

So we ask ourselves another question: why do we teach and learn Java as the first programming language?

Java is one of the most widespread and established languages in the world. It is used in industry, in large applications, and in enterprise systems. But its value lies not only in its widespread use: it lies in its structure.

Java is an object-oriented language. This means it teaches students how to model reality through concepts like classes and objects. In practice, it helps students represent the world in an abstract and organized way. It's not just about writing code, but about building mental models.

One of the most important aspects of Java, from an educational perspective, is its rigor. Java requires precision. It doesn't allow easy shortcuts. Every instruction must be clear, complete, and correct. This may seem like a limitation, especially at the beginning, but it's actually a great educational resource. It forces you to think before writing, to organize your ideas, and to follow a logic.

Furthermore, Java introduces fundamental concepts of modern computing, such as encapsulation, inheritance, and polymorphism. These words may seem complex, but they represent powerful tools for organizing your thoughts and managing complexity.

Another important element is modularity. In Java, code is organized into parts, modules, that can be reused. This teaches a fundamental principle: you don't start from scratch every time, but build on what already exists. This approach is valid not only in programming, but in every area of knowledge.

Of course, let's be honest: Java isn't the easiest language to get started with. The syntax can be complex, and compared to other, more straightforward languages, it can seem less intuitive. But this initial complexity is precisely what provides a training ground. Learning Java means training the mind for precision, discipline, and clarity.

This doesn't mean that teaching should be rigid or theoretical. On the contrary: to be effective, it must be concrete, engaging, and progressive. It's important to start with simple examples, related to students' real-world experiences, and gradually build more advanced skills. Practical projects, small games, and simulations are essential tools for making learning active and meaningful.

Java also offers a great advantage: it's a language that stays with students over time. What you learn in university holds true in research and the workforce. It's not just a teaching exercise, but a marketable skill.

Java isn't just a programming language. It's an educational tool. It's a training ground for thinking. It teaches you to be precise, to be rigorous, but also to be creative in finding solutions.

Perhaps this is precisely the point: teaching Java isn't about teaching how to write code. It's about teaching how to build solutions, how to organize your thoughts, and how to better understand the complex world we live in.

For the computer implementation of the analyzed algorithm, the programming language Java and the programming language Python was chosen.

Let's see below the source code related to the integration of Archimedes' parabola (which would be integrable exactly).

```
import java.util.Scanner;
import java.math.*;
public class IntegrazioneRettangolare {
   public static void main(String args[]) {
      Scanner tastiera = new Scanner(System.in);
      double a = -1.0, b = 1.0, N, h, k, x, y, AreaTot = 0.0, AreaRett;
      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

Programming is often thought of as learning to use a computer in an advanced way. In reality, it's much deeper. Programming means learning to think. It means taking a complex problem, breaking it down into simpler parts, finding a logical sequence of steps, and checking whether the solution works.

In other words, teaching programming means developing fundamental skills: logical thinking, analytical skills, problem-solving, but also patience and precision.

At this point, a second question arises: what second language can be used to begin this journey?

In recent years, Python has established itself as one of the most effective teaching tools. And that's 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.

When a student writes their first lines of code in Python, they don't immediately encounter a syntax barrier. They don't have to remember too many complex rules. Instead, they can quickly see the result of what they've written. This is a fundamental aspect: immediate feedback.

The ability to achieve results quickly increases motivation, reduces frustration, and makes learning more effective. The student doesn't feel stuck, but supported.

Another great value of Python is its versatility. It's not just an educational language: it's a real tool, used in a wide range of fields, from data science to artificial intelligence, from web development to automation.

This means that what is learned in school isn't confined to a theoretical exercise, but can be applied in concrete, current contexts.

From an educational perspective, Python has a decisive advantage: it reduces cognitive load. Students don't have to expend energy understanding complex syntax, but can focus on the problem at hand. This way, the focus of learning returns to thinking.

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 approach these skills, even without advanced technical training.

Naturally, it's important to be balanced. Python is not without limitations. Precisely because it is simple, it can sometimes obscure the complexity of its underlying mechanisms. If used without adequate guidance, it can lead to superficial learning.

For this reason, the role of the teacher is fundamental. Choosing the right language isn't enough: it's necessary to build an effective teaching path. A path that starts with concrete problems, that engages students, and that encourages experimentation.

It's helpful to propose practical activities, small projects, and real-world situations. For example, analyzing simple data, creating small games, or automating everyday tasks. This way, students immediately see the meaning of what they're doing.

Another important aspect is collaborative work. Programming isn't just an individual activity: it's also about sharing, comparing, and collectively building solutions. This develops fundamental transversal skills.

I'd also like to emphasize an often overlooked point: Python shouldn't be seen as a point of arrival, but as a starting point. It's an ideal tool for getting started, building confidence, and developing the foundations. These skills can then be transferred to other, more structured languages.

In this sense, Python and other languages, like Java, are not in competition, but complementary. Python opens the door, Java helps structure and consolidate.

I'd like to conclude this paraghaphwith a reflection.

Teaching Python isn't simply about teaching a programming language. It means offering students a tool to better understand reality, to tackle complex problems, and to develop critical and structured thinking.

In an increasingly complex world, these skills are essential. And Python, with its simplicity and power, represents one of the most effective ways to begin this journey.

For the computer implementation of the analyzed algorithm, also the programming language Python was chosen. Here is a function written in Python that is used to implement the rectangle method:

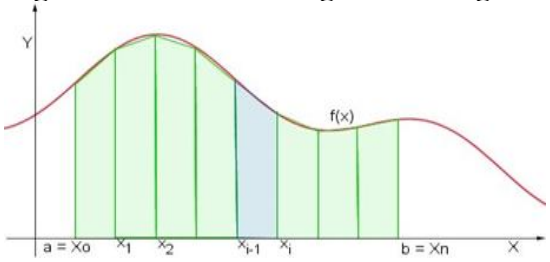
```
def rettangoli(f, a, b, n):
    dx = (b - a) / n
    area = 0
    for i in range(n):
        x = a + i * dx
        area += f(x) * dx
    return area
```

The trapezoidal method

Introduction

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

Figure 3. *numerical integration using the trapezoidal method*



Implementation in Java

Let's see below the source code related to 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

For the computer implementation of the analyzed algorithm, also the programming language Python was chosen. Here is a function written in Python that is used to implement the trapezoidal method:

```
def trapezi(f, a, b, n):
    dx = (b - a) / n
    area = (f(a) + f(b)) / 2
    for i in range(1, n):
        x = a + i * dx
        area += f(x)
    return area * dx
```

The method of parables

Introduction

As for approximate integration using the parabola method, the underlying idea is surprisingly simple, and also very elegant. If a curve is complicated, we can try replacing it, at least locally, with something simpler. In this case, we don't use a straight line, but a parabola.

Why a parabola?

Because a parabola is the simplest polynomial that can "follow" the curve of a curved function well. With three points, we can determine a parabola, and this parabola can approximate the function much more accurately 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 that passes 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.

The method of parabolas is a perfect example of how mathematics addresses complexity: not by always seeking exact solutions, but by building increasingly precise approximations.

This approach is fundamental not only in mathematics, but in all sciences. In physics, engineering, and economics, we often can't have an exact solution. But we can have a good enough solution.

And often, that's exactly what's needed.

From an educational perspective, the method of parabolas is incredibly valuable.

First, it bridges geometry and analysis. Students see concretely how a curve can be approximated by simpler shapes. It's not just a calculation, it's a visual representation.

Second, it introduces the concept of error. Every approximation involves an error, but this error can be controlled, reduced, and estimated. It's a fundamental idea: there's not just right or wrong, but also "how close to right."

Furthermore, this method develops algorithmic thinking. It's a procedure: divide, calculate, add. It's a repeatable, structured process, very similar to the way computers work.

And it's no coincidence that methods like this are the basis of numerical computation, that part of mathematics that allows computers to solve real-world problems.

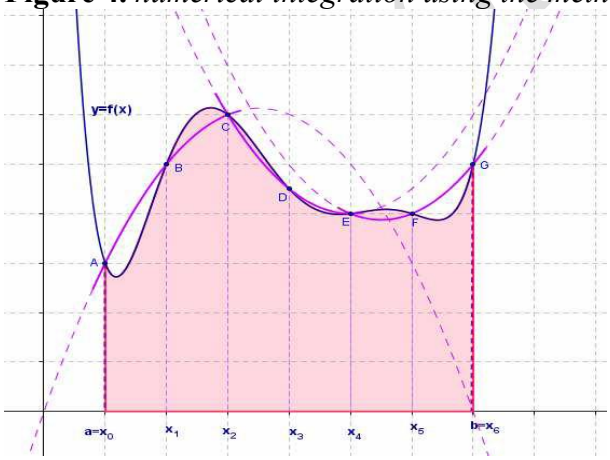
I'd also like to emphasize a cultural aspect. The parabola method demonstrates a living, dynamic, adaptable mathematics. It's not just a set of formulas to memorize, but a language for describing and addressing reality.

In conclusion, we can say that the parabola method isn't just a technique for calculating integrals. It's an example of how mathematics teaches us to think: to simplify without trivializing, to approximate without losing rigor, to find solutions even when perfection isn't possible.

And perhaps this is its most important lesson.

Together with the Rectangle and Trapezoid methods, it constitutes the set of Numerical Integration methods known as Newton-Cotes methods, whose characteristic is the subdivision 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 necessarily be an even number.

Figure 4. *numerical integration using the method of parables*



Let us therefore assume 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.

That is, let 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 trapezium methods, we discussed "construction of rectangles and trapeziums" 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 method, and by the function y=mx+p (a linear function) for the trapezium method.

Well, in Simpson's method, the graph of the integrand function y=f(x), present in two consecutive intervals, is replaced with the graph of the parabola that has three points in common with the graph of f(x); otherwise, the procedure is similar.

The final Cavalieri-Simpson formula for the entire integration interval [a,b], which allows us to calculate the definite integral of f(x)dx between the endpoints a and b, is as follows:

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

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

IT implementation

Together with the Rectangle and Trapezoid methods, it constitutes the set of Numerical Integration methods known as Newton-Cotes methods, whose characteristic is the subdivision 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 necessarily be an even number.

For educational purposes, it's best to let students do the implementation work in Java or Python. They can be provided with the initial portion of the program, which they will then have to complete.

As an example, the following is what can be given to the class:

```
// 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...
    …
```

Computer implementation in Python language

For the computer implementation of the analyzed algorithm, also the programming language Python was chosen.

Here is a function written in Python that is used to implement the parables method:

```
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

Introduction

Below, we would like to explore the educational aspects of this work in greater detail.

First, it is important to help students understand the geometric meaning of the integral defined as the area under a curve.

Furthermore, it is important to understand why, in many cases, the integral cannot be solved analytically and numerical methods are used.

Two methods of approximating the integral were examined and compared: rectangles and trapezoids.

It is important to encourage students to consider how the quality of the approximation depends on the choice of subdivision and the characteristics of the function.

The approach described in this work fosters the connection between visual and numerical reasoning, developing the relationship between abstract concepts and practical calculation.

The teaching activities described in this work have proven to be very useful for deepening both the mathematical and computer science aspects.

An interesting study to propose to students at this point could consist in studying the behavior of the approximation error as a function of the number of evaluations of the integrand function required.

For a long time, mathematics has been associated with the idea of a precise, elegant solution, enclosed in a formula. However, in real applications, things are often different. Many problems do not have an exact solution expressible in a simple form. Others are too complex to be solved with analytical methods.

And this is where numerical computation comes in.

Numerical computation is the branch of mathematics concerned with finding approximate solutions to complex problems, using algorithmic procedures and computational tools, often implemented on computers.

In other words, we don’t always seek the perfect solution: we seek a sufficiently precise solution, obtained efficiently.

We could say that numerical computation is the bridge between theoretical mathematics and the real world.

To better understand its importance, we can consider a simple example: the calculation of an integral. In many cases, it is not possible to find a primitive with classical techniques. Or consider the solution of nonlinear equations, the simulation of physical phenomena, or the prediction of economic models.

In all these cases, numerical computation becomes indispensable.

A fundamental aspect is that numerical computation is not limited to “doing calculations.” It introduces a new way of thinking: algorithmic thinking.

It means breaking down a problem into elementary steps, defining a procedure, and repeating it until an increasingly accurate result is obtained.

This approach is the basis of how modern computers work.

It’s 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.

Unlike pure mathematics, where absolute accuracy is sought, in numerical computation we accept the idea 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.

This is one of the most important ideas to understand: there is not only right and wrong, but also “how precise” is.

From an educational perspective, numerical computation has enormous value.

First, 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 idea: 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 Python, MATLAB, or other computing environments, 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.

A very significant example is the simulation of physical phenomena, such as the motion of planets, the diffusion of heat, or the behavior of fluids. In these cases, there are no simple formulas that describe everything exactly. Numerical calculation allows us to build approximate but extremely effective models.

I would also like to emphasize a cultural aspect.

Numerical calculation represents a paradigm shift: it is no longer just about solving problems, but about simulating them, exploring them, and understanding them through successive approximations.

It is a dynamic, experimental approach, very close to the scientific method.

In conclusion, we can say that numerical calculation is one of the most powerful tools of modern mathematics.

Not because it renounces precision, but because it accepts the complexity of the real world and addresses it with tools that are simultaneously flexible, efficient, and rigorous.

In a certain sense, numerical calculation teaches us an important lesson: it is not always possible to have a perfect answer, but it is always possible to get closer to the truth with ever greater precision.

And this, in science as in life, makes a big difference.

References

BERGAMINI MASSIMO, BAROZZI GRAZIELLA / MATEMATICA.BLU 2.0 2ED. - VOLUME 5 (LDM) / ZANICHELLI EDITORE, 2024

CAMAGNI PAOLO, NIKOLASSY RICCARDO / INFOM@T PER IL LICEO SCIENTIFICO OPZIONE SCIENZE APPLICATE / HOEPLI, 2023

https://www.treccani.it/enciclopedia/integrazione-numerica_%28Enciclopedia-della-Matematica%29/

https://it.wikipedia.org/wiki/Integrazione_numerica

Alfio Quarteroni / Matematica Numerica: Esercizi, Laboratori e Progetti / Springer-Verlag Italia, 2008

1 Aimé Mbobi / Principles and Applications of Numerical Integration Methods / LAP
2 LAMBERT Academic Publishing, 2012