Co-funded by the Erasmus+ Programme of the European Union

TOPIC PLAN		
Partner organization	Belgrade Metropolitan University	
Topic	Approximating Definite Integrals	
Lesson title	Definite integrals solving in Java	
Learning objectives	Student can interpret numerical integration rules. Student can compare two or more rules by their precision. Student can code Java algorithms for definite integrals solving.	**Methodology** x Modeling x Collaborative learning ☐Project based learning x Problem based learning **Strategies/Activities** ☐Graphic Organizer ☐Think/Pair/Share x Discussion questions **Assessment for learning** x Observations x Conversations x Work sample ☐Conference ☐Check list ☐Diagnostics **Assessment as**
Aim of the lecture / Description of the practical problem	Using a proper programming language it is possible to implement a broad family of algorithms for calculating the numerical value of a definite integral. The goal of this lesson is presentation of Java programming language in numerical integration problems.	
Previous knowledge assumed:	basics of mathematical analysis, basics of Java programming language, mathematical functions and expressions in Java	

Introduction / Theoretical basics	Definite Integral helps to find the area of a curve in a graph. It has limits, which are the start and the endpoints, within which the area under a curve is calculated. The limit points can be taken as [a, b], to find the area of the curve f(x), with respect to the x-axis. The corresponding expression of definite integral is: $\int_a^b f(x)dx$ Figure 1 Integration is the sum of the areas, and definite integrals are used to find the area within limits. Using a proper programming language it is possible to implement a broad family of algorithms for calculating the numerical value of a definite integral. This set of algorithms is known as numerical integration. The goal of this lesson is presentation of Java programming language in numerical integration problems. The basic problem in numerical integration is to compute an approximate solution to a definite integral (see Introduction - Figure 2) to a given degree of accuracy. If f(x) is a smooth function integrated over a small number of dimensions, and the domain of integration is bounded, there are many methods for approximating the integral to the desired precision, suche as: • the trapezoid rule; • Simpson's rule; • the midpoint rule; • and many others.	**learning** x Self-assessment ☐Peer-assessment x Presentation ☐Graphic Organizer x Homework **Assessment of learning** x Test ☐Quiz ☐Presentation x Project ☐Published work

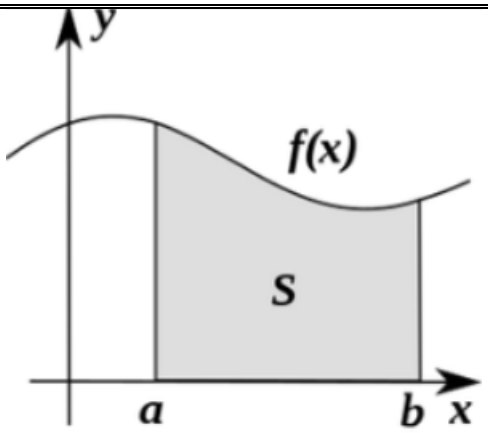

Figure 2

Numerical integration is used to calculate a numerical approximation for the value S that represent the area under the curve defined by f(x). A Definite Integral has start and end values: in other words there is an interval [a, b]. Finding the area enclosed by a graph of any function within the limits mentioned (say, [a, b]) on the graph is known as evaluating a definite Integral. In Definite Integrals, the limit goes from a to b (figure 3).

$$\int_a^b f(x) = F[a] - F[b]$$

Figure 3

Action

The Trapezoidal Rule in Java:

Java program will be named TrapezoidalRule.java and will contain the following functions:

- static double f(double x) - Standard normal distribution densityfunction and can be replaced with any sufficiently smooth function.
- static double integrate(double a, double b, int N) - Integrate f from a to b using the trapezoidal rule. Increasing N gives more precision result.

The first function is exposed by the following snippet:

```
static double f(double x) {
```

```
    return Math.exp(- x * x / 2) / Math.sqrt(2 * Math.PI);
  }
```

The snippet of an integration function is given further:

```
static double integrate(double a, double b, int N) {
    double h = (b - a) / N;             // step size
    double sum = 0.5 * (f(a) + f(b));    // area
    for (int i = 1; i < N; i++) {
      double x = a + h * i;
      sum = sum + f(x);
    }

    return sum * h;
  }
```

The result is as follows:

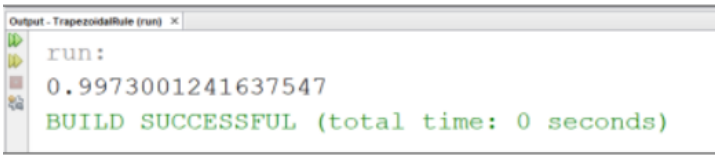

Figure 4

The software answer can be considered good enough because the real answer is: 0,9973002040.

Simpson's Rule in Java

Java program will be named SimpsonsRule.java and will contain the following functions:

- static double f(double x) - Standard normal distribution densityfunction and can be replaced with any sufficiently smooth function.
- static double integrate(double a, double b, int N) - Integrate f from a to b using the trapezoidal rule. Increasing N gives more precision result.

The first function is exposed by the following snippet:

```
public static double f(double x) {
    return Math.exp(- x * x / 2) / Math.sqrt(2 * Math.PI);
  }
```

The snippet of an integration function is given further:

```
public static double integrate(double a, double b) {
    int N = 10000;                  // precision parameter
    double h = (b - a) / (N - 1);     // step size
```

```
    // 1/3 terms
    double sum = 1.0 / 3.0 * (f(a) + f(b));

    // 4/3 terms
    for (int i = 1; i < N - 1; i += 2) {
      double x = a + h * i;
      sum += 4.0 / 3.0 * f(x);
    }

    // 2/3 terms
    for (int i = 2; i < N - 1; i += 2) {
      double x = a + h * i;
      sum += 2.0 / 3.0 * f(x);
    }

    return sum * h;
  }
```

In main() function, the integration will be tested over interval [-3, 3]. This differs a bit comparing to previous case:

```
public static void main(String[] args) {
    double a = -3;
    double b = 3;
     System.out.println(integrate(a, b));
  }
```

The result is as follows:

Figure 5

The software answer can be considered good enough because the real answer is: 0,9973002040.

Individual task:

Use this link and choose arbitrary smooth function: https://www.calculushowto.com/smooth-function/. For

	the selected function, create your own software solution for implemetation of: • Trapezoidal Rule; • Simpson's rule. 1. Run and demonstrate results of created software solution. 2. Compare your and real result and calculate error. 3. Which solution is more precise? **Homework:** Use this link and choose arbitrary smooth function: https://www.calculushowto.com/smooth-function/. Use next link to study the application of the Midpoint Rule in Java: http://theflyingkeyboard.net/java/java-midpoint-rule-rectangle-method/. For the selected function, ***create your own software solution for implemetation of the Midpoint Rule in Java***. 1. Run and demonstrate results of created software solution. 2. Compare your and real result and calculate error. 3. Compare this result with rezults of teh Trapezoidal Rule and Simpson's rule. 4. Which solution is the most precise?	
Materials / equipment / digital tools / software	*The materials for learning* are given as a part of references of the end from this topic plan; *Equipment*: classroom, board, chalk; *Digital tools*: laptop, projector, software tools as Java JDK 8(or above), Netbeans IDE 8.2 (or above);	

Consolidation	• The teacher's discussion with the students through appropriate questions; • Independent solving of simple tasks by the students under the supervision of the teacher; • Given of examples by the teacher for introducing a new concept in a cooperation and a discussion with the students; • Assignment of homework by the teacher with a time limit until the next class.

Reflections and next steps

Activities that worked	**Parts to be revisited**

References

1. Dr Vladimir Milićević, Elektronski materijali predavanja za učenje, Metropolitan Univerzitet, 2021 - 2022. godina, Beograd
2. https://www.cuemath.com/calculus/definite-integral/
3. https://en.wikipedia.org/wiki/Numerical_integration#Methods_for_one-dimensional_integrals
4. https://www.mathsisfun.com/calculus/integration-definite.html
5. https://math.libretexts.org/Courses/Mount_Royal_University/MATH_2200%3A_Calculus_for_Scientists_II/2%3A_Techniques_of_Integration/2.5%3A_Numerical_Integration_-_Midpoint%2C_Trapezoid%2C_Simpson%27s_rule
6. http://www.math.pitt.edu/~sparling/23021/23022numapprox2/node4.html
7. https://en.wikipedia.org/wiki/Trapezoidal_rule
8. https://www.cuemath.com/simpsons-rule-formula/
9. https://introcs.cs.princeton.edu/java/93integration/
10. https://www.calculushowto.com/smooth-function/
11. http://theflyingkeyboard.net/java/java-midpoint-rule-rectangle-method/
12. Y. Daniel Liang, Introduction to Java Programming, Comprehensive Version, 10th edition