TP C#10 : Mathmageddon

Assignment

Archive

You must submit a zip file with the following architecture :

```
rendu-tp-firstname.lastname.zip
|-- firstname.lastname/
    |-- AUTHORS
    |-- README
    |-- TP10/
        |-- TP10.sln
        |-- TP10/
            |-- Everything except bin/ and obj/
```

Don't forget to check the following points before submitting :
- You will obviously replace *firstname.lastname* with your login, and don't forget the AUTHORS file.
- The AUTHORS and README files are mandatory.
- Do not leave *(*bin) and *(*obj) files in your project.
- You must follow scrupulously the required function prototypes.
- You must remove every test file from your code.
- *(*Your code must compile !)

AUTHORS

This file must contain the following : a star (*), a space, your login and a newline.
Here is an example (where '$' represents the newline and ' ' a whitespace) :

```
* firstname.lastname$
```

Please note that the filename is AUTHORS with NO extension. To easily create your `AUTHORS` file, you can type the following command in a terminal :

```
echo "* firstname.lastname" > AUTHORS
```

README

In this file, you can write any comments about the practical, your work, or more generally about your strengths / weaknesses. You must list and explain all the boni you have implemented. An empty `README` file will be considered an invalid archive (malus).

1 Introduction

1.1 Objectives

Last week, you played with physics. This week, we are going to link another course to programming : mathematics, and more precisely its analytical domain. It is good to know how to write formulas and compute them on a sheet of paper, but it would be even better to know how to code main aspects of the subject as derivatives and integrals of any function. You are going to learn during this practical to approximate those values, and to reduce the induced error margin to get the optimal accuracy. These are not easy notions, but if you understand this practical you will be better at applying programming to your surroundings.

2 Courses

Before starting this lecture, you need to understand that it is required to read it carefully. You may lose lots of time if you do not, as it contains useful information that you will need to use. If you don't understand something, don't hesitate to ask assistants for help.

2.1 Derivatives

Let us begin by redefining the basics. Before speaking about function derivatives, we will start by the derivative of the function at one point.

Definition

The derivative of a function f of a single variable x at a chosen input value, when it exists, is the slope of the tangent line to the graph of the function at the point$(x, f(x))$. This tangent line is the best linear approximation of the function near that input value.

Thus, the derivative of a function f is a function that, for all numbers for which f admits a derivative number, associates this derivative number.

When deriving functions, most of the time we use general formulas like $(sin(x))' = cos(x)$. Even if it is technically possible to do the same in computer science – with string management, parsing, and more – it is not the point of this practical. Here, we will not try to find the general formula of the derivative of a function, but simply compute its value in targeted points. By computing a lot of close points on a grid, we will be able to see something that looks like the derivative of f, the remaining points being approximated.

You have all learned the following formula, even if some of you chose to forget it.

Formula

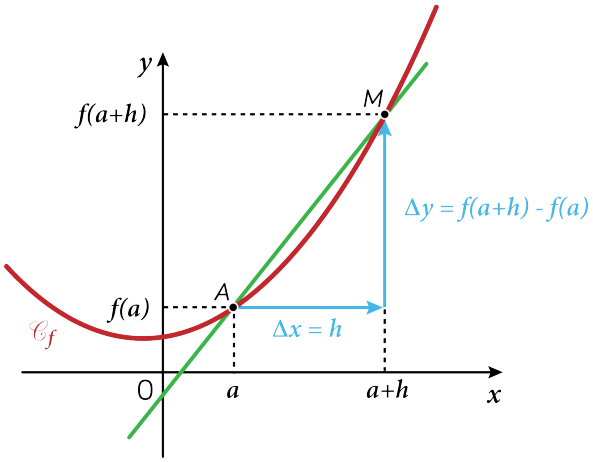
Let f be a function defined on the interval I of $\mathbb{R}$, and $a \in I$. Let h a real number not null such as $a + h \in I$. We call **rate of change** of the function f between a and $a + h$, the real number

$$\frac{f(a+h) - f(a)}{h} = \frac{\Delta y}{\Delta x}$$

As an example, the rate of change of $f : x \to x^2$ between 1 and $1 + h$ is

$$\frac{f(1+h) - f(1)}{h} = \frac{(1+h)^2 - 1^2}{h} = \frac{2h + h^2}{h} = 2 + h$$

Here is a simple visual to better understand the rate of change.



NB

If you encounter Δ in math, this generally means those values are meant to vary.

From the rate of change, we can define the formula of the derivative of a function at one point.

Formula

Let f a function defined on an interval I of $\mathbb{R}$, and $a \in I$. We state that **the function is differentiable on a** if the rate of change $f'(a)$ when h tends to 0 is

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

The number $f'(a)$ – when it exists – is the slope of the tangent to the graph at the point of abscissa a.

We all know the derivative of $f(x) = x^2$. It is $f'(x) = 2x$. We can easily see the underlying relation. Given the latter computation : if h is null, then the rate of change from 1 to 1 (thus at $x = 1$) is 2. Plus we know that $f'(1) = 2$. Thus, theoretically as well as in practice, we get the result when h is equal to 0.

From mathematics to programming, you need to know how to translate formulas that have a limit. A human easily gets the level of abstraction required to understand a limit, but a computer cannot. In other words, it is impossible to code such a formula.

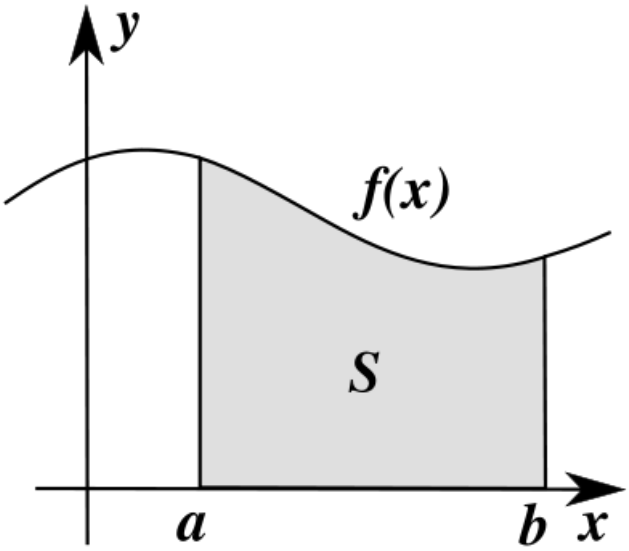
Hence in programming, if the exact formula is not known all one can do is approximate the derivative, by assigning to h arbitrary values close to zero. This choice leads to a **margin of error** that can be controlled according to one's needs.

2.2 Integrals

Integration is, as you may know, the opposite operation of differentiation. But one question remains. What can you do with them ? Once again, mathematics are a tool, and the last practical should have given you clues of their importance to solve various problems, in mechanics for example.

To find the approximation of the derivative, we started from the formula. But for integrals it is not possible to do this. It is a lot easier to refer to the geometrical / graphical point of view to find an approximation of the studied integral.

As you must know, the integral of a function between a and b is simply the area under the graph - here S.



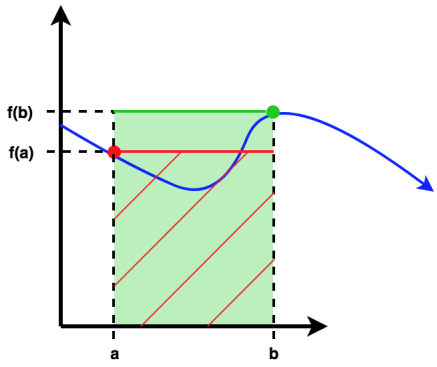
There exist many methods that permit one to estimate the value of the integral of a function.

Attention

We focus on defined integrals, which means always on an interval from a to b with $a \leq b$

2.2.1 Rectangle rule

The rectangle rule is also known as the *one-point method*. It consists of the approximation of the area under the graph of $f(x)$ by a rectangle. You can get this rectangle by drawing the horizontal segment starting from one of the two points of the graph. The segment then links the couple of points $\{(a, f(a)), (b, f(a))\}$ (red rectangle) or $\{(a, f(b)), (b, f(b))\}$ (green rectangle).



Formula

The area of the lower rectangle (the red one) is

$$\int_a^b f(a)d(x) = b * f(a) - a * f(a) = f(a) * (b - a)$$

The area of the upper rectangle (the green one) is $\int_a^b f(b)d(x) = b * f(b) - a * f(b) = f(b) * (b - a)$.

It is difficult to miss that this approximation method, as presented in the picture, critically lacks precision. However, we can strongly enhance the result by dividing the studied interval in a multitude of equal segments. The closer the two points are, the better the approximation of the actual value.

Let's then split an interval $I = [a, b]$ into n intervals of same length $\frac{(b-a)}{n}$. The integral of $f(x)$ on this interval can be computed as the sum of the integrals of each interval.

Formula

The area of $f(x)$ on I is

$$\int_a^b f(x)dx = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + ... + \int_{x_{n-1}}^{x_n} f(x)dx$$

We replace the function $f(x)$ by constant functions, containing one of the two points of the interval $[x_i, x_{i+1}]$, easily integrable - the area being reduced to a rectangle - which results in two approximations :

Formula

lower rectangles :

$$A_1 = \int_{x_0}^{x_1} f(x_0)dx + \int_{x_1}^{x_2} f(x_1)dx + ... + \int_{x_{n-1}}^{x_n} f(x_{n-1})dx$$

$$A_1 = \frac{b-a}{n} \sum_{i=0}^{n-1} f(x_i)$$

upper rectangles :

$$A_2 = \int_{x_0}^{x_1} f(x_1)dx + \int_{x_1}^{x_2} f(x_2)dx + ... + \int_{x_{n-1}}^{x_n} f(x_n)dx$$

$$A_2 = \frac{b-a}{n} \sum_{i=1}^{n} f(x_i)$$

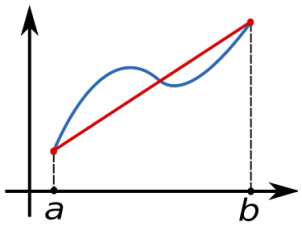
Of course, $x_0 = a$ et $x_n = b$.

Note

The mean value of those two computations is actually equal to the approximation of the integral $\int_a^b f(x)dx$ by the trapezoidal rule.

2.2.2 Trapezoidal rule

The trapezoidal rule is also known as the *two-points method*. It is about approximating the area below the graph of the function $f(x)$ by a trapezoid and calculating its area. To achieve this, the method uses the two points of abscissa a and b, by connecting them to make a segment of points $(a, f(a))$ and $(b, f(b))$ (picture below). This segment is the upper side of the trapezoid.



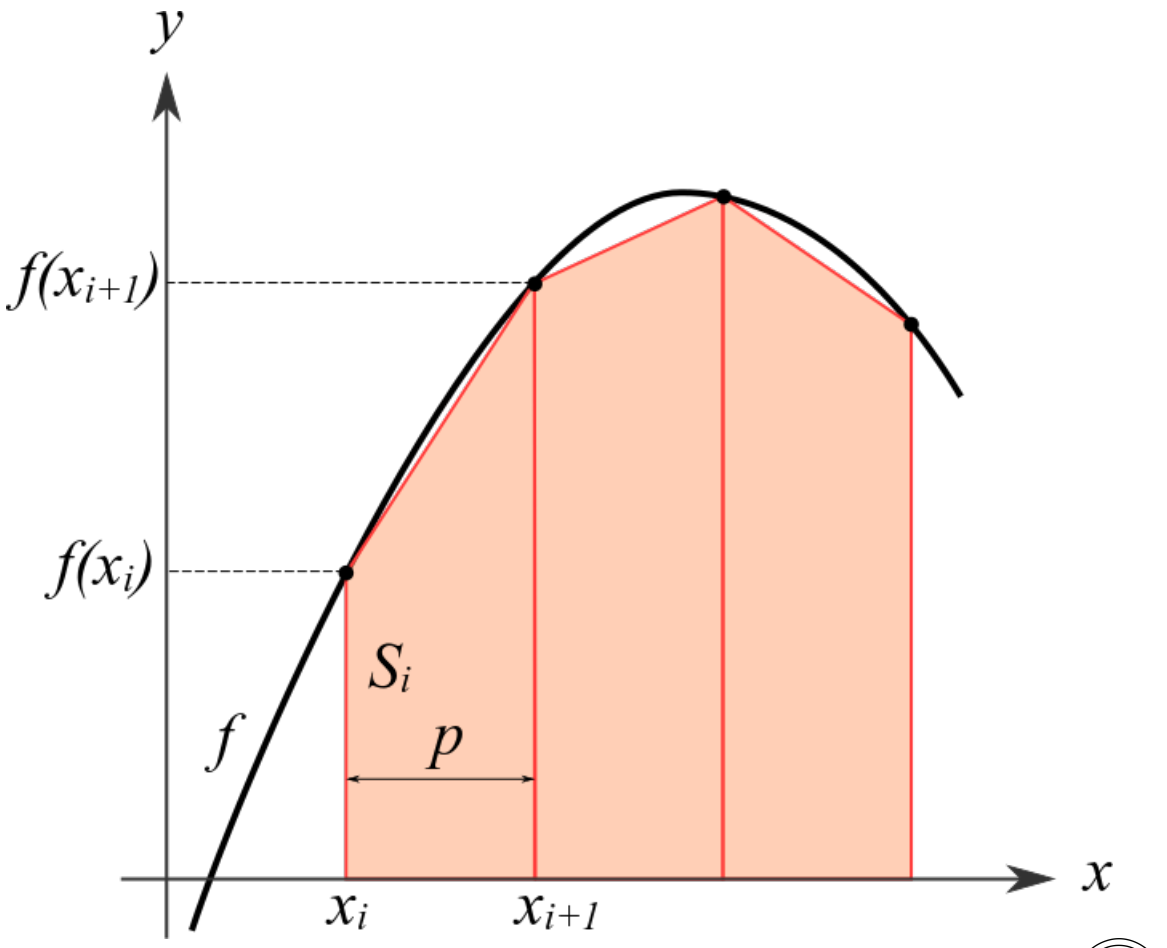
Again, we can easily see why the best this method can do is return a raw approximation. It approximates the function by a line connecting two points that can be far from each other. However, the closer the points are to each other, the smaller the error margin.

Formula

The area under the graph is approximately equal to the area of the trapezoid.

$$\int_a^b f(x)dx \approx (b-a)[\tfrac{f(a)+f(b)}{2}]$$

In the same way as the rectangle rule, the approximation is enhanced by the studied interval's shrinking.



Thus, we can define a new formula :

Formula

Let $\{x_k\}$ an interval of $[a, b]$ such as $a = x_0 < x_1 < ... < x_{N-1} < x_N = b$ then

$$\int_a^b f(x)dx \approx \sum_{k=1}^{N} \frac{f(x_{k-1}) + f(x_k)}{2}(x_k - x_{k-1})$$

This formula looks heavy but don't worry, it will not be used in that state. This formula is valid for sub-intervals of possibly different sizes. By choosing to focus on sub-intervals of same size, it can be greatly simplified. We then have :

Formula

Let $\Delta x_k = \Delta x = \frac{b-a}{N}$ with N the number of intervals of same size then

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2} \sum_{k=1}^{N} (f(x_{k-1}) + f(x_k))$$

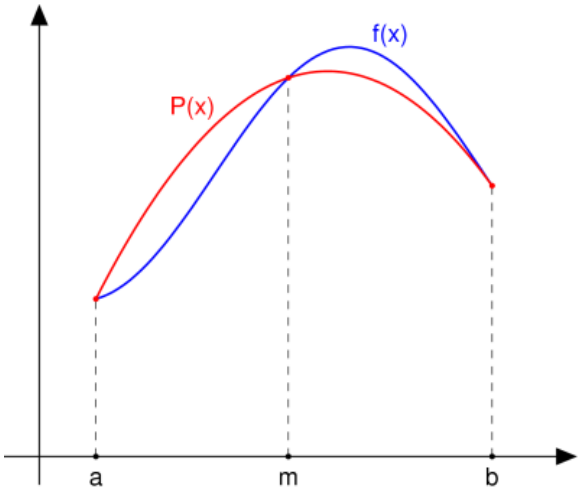
2.2.3 Simpson's rule

Simpson's rule is also known as the *three-points method*. The main fault of the rectangle or trapezoidal rules is that they lack finesse when approximating graphs. The Simpson rule can be compared to the former but will approximate the function by a quadratic polynomial. This polynomial is defined by three points of abscissas a, b, and $m = \frac{a+b}{2}$ having respectively the same ordinates as the f curve.

Formula

By choosing the points $(a, f(a))$, $(b, f(b))$, $(m, f(m))$, with $m = \frac{a+b}{2}$, the researched polynomial can be written by Lagrange interpolation

$$P(x) = f(a)\frac{(x-m)(x-b)}{(a-m)(a-b)} + f(m)\frac{(x-a)(x-b)}{(m-a)(m-b)} + f(b)\frac{(x-a)(x-m)}{(b-a)(b-m)}$$



The polynomial is a very simple function to integrate, and the error margin is far inferior to those of the previous methods.

Formula

The area under the graph is approximately equal to the one of the polynomial.

$$\int_a^b f(x)dx \approx \frac{b-a}{6}[f(a) + 4f(\frac{a+b}{2}) + f(b)]$$

Once again, the smaller the interval, the better the approximation. Let's introduce the composite Simpson formula.

Formula

Let n sub-intervals in $[a, b]$ with n even, $h = \frac{b-a}{n}$ their length, and $x_i = a + ih$ for i an integer from 0 to n,

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(x_0) + 2\sum_{j=1}^{\frac{n}{2}-1} f(x_{2j}) + 4\sum_{j=1}^{\frac{n}{2}} f(x_{2j-1}) + f(x_n)]$$

NB

The presented Simpson rule is the usual rule, also known as 1/3 rule. The 3/8 Simpson rule uses a third degree polynomial and its formula is a lot heavier.

2.2.4 Newton-Cotes rule

Newton-Cotes rule is the generalization of all the previous methods. It is about choosing equidistant points to create sub-intervals of the same length. In other words, by using this method with only one point, it is similar to using the rectangle method, with 2 points it gets to the trapezoidal rule, and 3 points leads to Simpson's rule.

The goal of this formula is to create by lagrange interpolation a polynomial of degree n, easily computed.

Formula

Let $x_i = a+i\Delta$, for $i = 0, ..., n$, $\Delta = (b-a)/n$ and $y = (x-a)/\Delta$. The nth degree formula can be written as

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i)$$

with

$$w_i = \frac{b-a}{n}\frac{(-1)^{n-i}}{i!(n-i!)}\int_0^n \prod_{k=0,k\neq i}^{n}(y-k)dy$$

As you can see, the formula to compute w_i contains an integral, so we will need to use an approximation. Here is a small chart containing some w_i computed for you.

Degree (n)	Number of Points	Name
1	2	Trapezoidal rule
2	3	1/3 Simpson's rule
3	4	3/8 Simpson's rule
4	5	Boole-Villarceau's rule
6	7	Weddle-Hardy's rule
7	8	8 points method
8	9	9 points method
9	10	10 points method
10	11	11 points method

Degree (n)	w (from 0 to n)	constant
1	1, 1	$\frac{b-a}{2}$
2	1, 4, 1	$\frac{b-a}{6}$
3	1, 3, 3, 1	$\frac{b-a}{8}$
4	7, 32, 12, 32, 7	$\frac{b-a}{90}$
6	41, 216, 27, 272, ...	$\frac{b-a}{840}$
7	751, 3577, 1323, 2989, ...	$\frac{b-a}{17280}$
8	989, 5888, -928, 10496, -4540 ...	$\frac{b-a}{28350}$
9	2857, 15741, 1080, 19344, 5778, ...	$\frac{b-a}{89600}$
10	16067, 106300, -48525, 272400, -260550, 427368, ...	$\frac{b-a}{598752}$

The integration of Simpson's rule polynomial is a good example of the use of the above values. In a more generic way, we have :

Formula

Let n sub-intervals in $[a, b]$ with n even, $h = \frac{b-a}{n}$ their length, and $x_i = a + ih$ for i an integer from 0 to n,

$$\int_a^b f(x)dx \approx < constant > [w_0 f(x_0) + w_1 f(x_1) + ... + w_n f(x_n)]$$

2.2.5 Gauss-Legendre quadrature

Let's jump to the Gauss-Legendre quadrature. Generally, a quadrature is an approximation of the definite integral of a function, usually stated as a weighted sum of function values at specified points within the domain of integration. The Gaussian quadrature rule is a method constructed to yield an exact result for polynomials of degree $2n-1$ with n points taken on the integration domain.

Formula

For an interval $[a, b]$,

$$I = \int_a^b f(x)\varpi(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

where $\varpi : (a, b) \to \mathbb{R}_+$ is a weighting function, that can assure the integrability of f.

In this practical, we will focus on the variant of this rule, called Gauss-Legendre's rule. It can only resolve the most classical integration problems. It is about integrating the function f on $[a, b]$. The n nodes are the roots of the nth polynomial of Legendre $P_n(x)$.

Formula

Coefficients are given by

$$w_i = \frac{-2}{(n+1)P'_n(x_i)P_{n+1}(x_i)} = \frac{-2}{(1-x_i^2)P'_n(x_i)^2}$$

Here is a small chart with formulas

Number of points, n	Weight (w_i)	Points (x_i)	Legendre polynomial
1	2	0	x
2	1, 1	$\check{}\sqrt{1/3}$,$\sqrt{1/3}$	(3x2 – 1)/2
3	5/9, 8/9, 5/9	$\check{}\sqrt{3/5}, 0$,$\sqrt{3/5}$	(5x3 – 3x)/2

A small example for a simple integration

$$\int_{-1}^{1}(x+1)^2dx = 1(\frac{1}{\sqrt{3}}+1)^2 + 1(-\frac{1}{\sqrt{3}}+1)^2 = \frac{8}{3}$$

and we can easily verify this because we know the primitive of $(x+1)^2$

$$\int_{-1}^{1}(x+1)^2 = \left[\frac{(x+1)^3}{3}\right]_{-1}^{1} = \frac{8}{3}$$

3 Error margin

One of the goals of this practical is to show you that all these rules seem quite similar but do not have the same results. They all compute the same things but with varying degrees of accuracy or precision. We have presented these methods from the lesst to the most precise.

Formula

Basically, the formula to know the error margin (δ) is

$$\delta = \int_a^b f(x)dx - \text{'the value that you found for the calculus of the same integral'}$$

For the exercises of precision required for each method, we will need to use this computation according to the number of sub-intervals chosen between a and b.

4 Finally, some code.. Or is there ?

4.0.1 Delegates and lambda expressions

We are going to use some nice features that you have not seen yet, such as delegates and lambda expressions. As these are complicated notions, this is just an introduction.

The main use of lambdas expressions is to generically be able to pass a mathematical function as an argument of a function. Here is how we declare it :

```
1  Func<double, double> f = x => Math.exp(x);
```

This line creates a Func variable (a.k.a delegate), with the name f. We remark angle brackets after the keyword Func : they indicate that the function takes a double as input and returns a double in output. The right part is a lambda expression ; the line is apprehensible in itself. For each x given, the line will transform the input x into $exp(x)$. Which is exactly the definition of exponential of x.

Thus, we can do things like :

```
Func<double, double> f = x => Math.Exp(x);
Func<double, double> g = x => Math.Log(x);
Func<double, double> h = x => x * x;
Console.WriteLine(f(g(2)) + h(4)); // 18

// with
public double example(Func<double, double> f)
{
    return f(2);
}

Console.WriteLine(example(h)); // 4
```

4.0.2 Dictionaries

Why not go on to the dictionaries ? These are data structures often used when needing to apply the same notions to a big collection of objects. A dictionary is composed of a pair $key, value$. The key allows to access to its $value$ by using it with the dictionary.

```
// declaration
Dictionary<string, string> profs = new Dictionary<string, string>();

// examples
profs.Add("Prof1","Krisboul");
profs.Add("Prof2", "Junior");
profs.Add("Prof3", "ChoKaPeek");

Console.WriteLine(profs["Prof1"]); // "Krisboul"
profs.Remove("Prof3"); // remove "ChoKaPeek" from the dictionnary
```

Aaand, it's done. Finally... There is a lot of information on this course, do not hesitate to read it again, or ask assistants for help. Now, let's have fun.

5 Exercises

5.0.1 Derivative

Let us start simply. Code the function that takes a point a, the value h and a lambda expression and returns the rate of change of the lambda. Please do this with 6 decimals after the comma. Also, do not forget to handle possible exceptions!

```
public static double RateOfChange(double a, double h,
    Func<double, double> f);
```

Afterwards, we need a function that generates a list of points from the rate of change in the interval $[a, b]$ and with increment t.

```
public static List<double> GeneratePointsForPlot(double a, double b,
    double t, Func<double, double> f);
```

5.1 Integrals

5.1.1 Rectangle Rule

Start with a function that calculates the integral between a and b with the lower rectangle rule.

```
public static double IntegralRectangleRule(double a, double b,
    Func<double, double> f);
```

Now, split the interval into small intervals of size n and use the composite rectangle rule to find the integral.

```
public static double CompositeIntegralRectangleRule(double a, double b,
    double n, Func<double, double> f)
```

The most important part is to find the error margin of such a technique. You will have to know the actual integral of the function you will test so please use simple functions such as $sin(x)$ or $exp(x)$.

```
public static double CIRRErrorMargin(double a, double b,
    double n, Func<double, double> f, Func<double, double> F)
```

Try to play around with functions and values to get an idea of how efficient the rule is.

5.1.2 Trapezoidal Rule

Start with a function that calculates the integral between a and b with the rectangle rule.

```
public static double IntegralTrapezoidalRule(double a, double b,
    Func<double, double> f);
```

Now, split the interval into small intervals of size n and use the composite trapezoidal rule to find the integral.

```
public static double CompositeIntegralTrapezoidalRule(double a, double b,
    double n, Func<double, double> f)
```

The most important part is to find the error margin of such a technique. You will have to know the actual integral of the function that you will test so please use simple functions such as $sin(x)$ or $exp(x)$.

```
public static double CITRErrorMargin(double a, double b,
    double n, Func<double, double> f, Func<double, double> F)
```

Try to play around with functions and values to get an idea of how efficient the rule is. What happens to the error margin of an affine function ?

5.1.3 Simpson's Rule

Start with a function that calculates the integral between a and b with Simpson's rule.

```
public static double IntegralSimpsonRule(double a, double b,
    Func<double, double> f);
```

Now, split the interval into small intervals of size n and use the composite Simpson rule to find the integral.

```
public static double CompositeIntegralSimpsonRule(double a, double b,
    double n, Func<double, double> f)
```

The most important part is to find the error margin of such a technique. You will have to know the actual integral of the function you will test so please use simple functions such as $sin(x)$ or $exp(x)$.

```
public static double CISRErrorMargin(double a, double b,
    double n, Func<double, double> f, Func<double, double> F)
```

Try to play around with functions and values to get an idea of how efficient the rule is. What happens to polynomials of degree 2 ?

5.1.4 Newton-Cotes Rule

Write a function that finds the integral of f in the interval $[a, b]$ with the Newton-Cotes rule. Use 6 points. You must start by finding your constants, keep in mind that 6 points means degree 5.

```
public static double IntegralNewtonQuadratureAt6(double a, double b,
    Func<double, double> f);
```

Now, please apply the previous function to find the Newton-Cotes approximation. Here, n is the degree we will choose, so you will need to create a dictionary to store the values of w_i. You should implement it in the scope of your class.

```
public static void InitDictionary();

public static double IntegralNewtonQuadrature(double a, double b,
    int n, Func<double, double> f)
```

Now, please apply the previous function to find the Composite Newton-Cotes approximation. Here, n is the size of the intervals we will choose to divide the interval $[a, b]$ and d is the degree.

```
public static double CompositeIntegralNewtonQuadrature(double a, double b,
    double n, int d, Func<double, double> f)
```

The most important part is to find the error margin of such a technique. You will have to know the actual integral of the function you will test so please use simple functions such as $sin(x)$ or $exp(x)$.

```
public static double CINQErrorMargin(double a, double b,
    double n, int d, Func<double, double> f, Func<double, double> F)
```

Please try to play around with functions and values to get an idea how efficient the rule is. What happens to polynomials of degree n ? Why is this method rarely used for degrees upper than 7 ?

5.1.5 Gauss-Legendre Quadrature (Bonus)

This part will be harder. However, hang on and try to do it. Previous exercises are short, you have plenty of time to do the bonus. It is important to notice that here, n will be the number of points on the studied interval.

The main function is :

```
public static double IntegralGaussQuadrature(double a, double b,
    uint n, Func<double, double> f)
```

Can you show in which ways the Gauss-Legendre quadrature is better than the rules previously seen ?

These violent deadlines have violent ends !