https://doi.org/10.53656/math2023-4-4-int

INTEGRATED LESSONS IN CALCULUS USING SOFTWARE

Dr. Pohoriliak Oleksandr, Assoc. Prof.[1)] Dr. Olga Syniavska, Assoc. Prof.[1)]
Dr. Anna Slyvka-Tylyshchak, Assoc. Prof.[1)] Dr. Antonina Tegza, Assoc. Prof.[1)]
Prof. Dr. Alexander Tylyshchak[2)]
[1)]Uzhhorod National University – Uzhhorod (Ukraine)
[2)]Ferenc Rákóczi II Transcarpathian Hungarian Institute – Berehove (Ukraine)

Abstract. Today, both secondary and higher education cannot be effective without the use of innovative technologies. Therefore, all pedagogical workers try to use modern means of organizing educational activities, which cover the entire learning process from defining the goal to reaching the results.This article proposes to consider one of the methods of conducting integrated classes with a combination of analytical computing and specific software application. The above considerations can help students more effectively master the complex theoretical material of the calculus course. The article aims to consider some methods of studying and interpreting the theory of calculus using the free computer mathematics system Maxima and the Python language.

Keywords: calculus; computer mathematics system; Maxima; Python; differential calculus; integration

1. Introduction

Twenty years ago, many ministries of education put forward the following theses to teachers: "The main task of the 21st century is the modernization of education. Education must acquire an innovative character. Information technologies open up new opportunities for knowledge of human activities, etc." Since then, many changes and amendments have been made to the curricula and programs of secondary and higher schools. However, there are still many discussions on the development of methodological concepts and approaches to studying, first, subjects of the natural and scientific cycles. In particular, this also applies to modern mathematics, since it already acts not only as a branch of knowledge but also as a powerful method of scientific knowledge in other sciences.

The use of innovative technologies in the educational process is already an integral part of the modern world. This is especially true when teaching fundamental disciplines for IT specialties. For many teachers several years of experience show that conducting integrated classes with a combination of analytical calculations and software helps students master the complex theoretical material of higher mathematics.

This article discussed using the Maxima software environment and the Python programming language to study the course of theoretical and practical calculus effectively. A computer algebra system (CAS) is a multifunctional software tool that can perform mathematical operations with data in symbolic or numerical form and create visualizations of data and calculation results. Using CAS, it is possible to avoid cumbersome calculations when solving typical mathematical problems, such as calculating the values of functions, solving equations and inequalities and their systems, and constructing various graphs. Commercial programs such as Maple, Wolfram Mathematica, MATLAB, MathCad, etc. are among the most well-known CASs, and Axiom, FriCAS, Maxima, Reduce, etc. are freely distributed CASs. However, when studying mathematical disciplines, students should prefer using free programs, specifically Maxima[1]. Maxima is a universal CAS that can solve various mathematical problems. For example, this program allows performing various transformations of expressions, such as

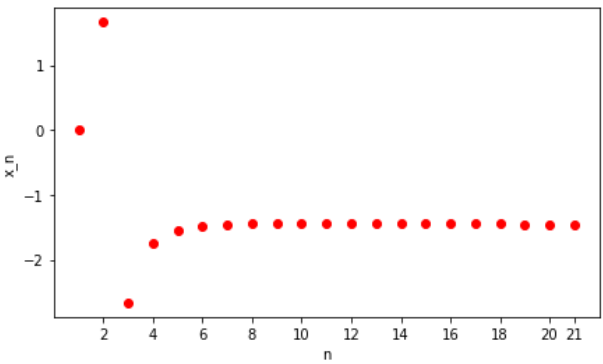

Figure 1

```
In [1]: Eps=float(input('Enter the accuracy Eps='))
        def fun(n):
            return  (1+1/n)**n
        i=1
        while abs(fun(i)-fun(i+1))>Eps:    i+=1
        print('Result:')
        print('n=', i+1)
        print('e=',fun(i+1))
```

```
Enter the accuracy Eps=0.00000001
Result:
n= 11657
e= 2.7181652432261854
```

Figure 2

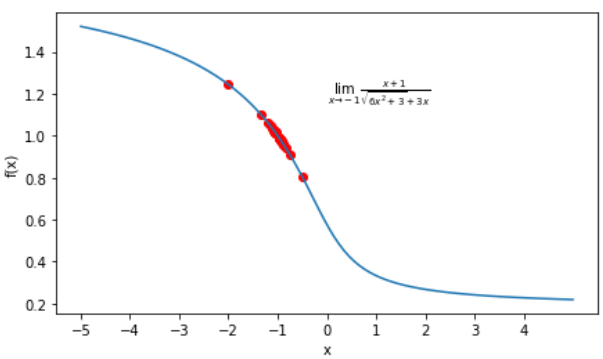

Figure 3

simplification, substitution, graphing, finding limits of functions, differentiation, calculating indefinite and definite integrals, expansion of functions into series, etc. (De Souza et al. 2004). Among the main advantages of using Maxima[1], in addition to open access, is a user-friendly interface with built-in menus for basic mathematical operations. Also, a significant advantage is the availability of versions of Windows, Mac OS X, Linux, and Android[1]. The issues of using the Maxima system in solving calculus problems have been studied by many authors, in particular (Fauzi et. al. 2012); (Karjanto 2021); (Karjanto & Husain 2021).

An additional tool for visualization and practical implementation of the theory of mathematical analysis is the object-oriented programming language Python[2]. Python has a set of structured programming constructs and contains many libraries for scientific calculations. A significant advantage over other programming environments is that Python is available on various platforms, including Linux. The use of Python in teaching calculus was considered in articles (Pereira Junior et al. 2022); (Park et al. 2019); (Vanderplas & Schanafelt 2017).

2. Aim of the study

This article proposes to raise the question of the methodology of teaching the theoretical course of mathematical analysis so that its study is more accessible and exciting to students and rationally combines the logical rigor of the discipline and its visibility. In this article, we focus only on some aspects of the mathematical analysis course and consider visualization of some theoretical concepts, statements, and formulas of calculus, as well as how to solve practical tasks using various application programs.

3. Results and Discussion

Limits of Sequences and Functions using Python. The course of calculus usually starts with studying the theory of real numbers. However, since students easily understand this topic, we do not focus on it. The next step is to study the theory of numerical sequences. For the student to better learn the concepts and properties of infinitely small and large numerical sequences, it is worth considering tasks of various types. Thus, along with studying various purely technical methods of calculating limits, it is advisable to offer students the task of visualizing the convergence of the sequence in the figure and calculating the limit with a given accuracy.

Example 1. Evaluate $\lim\limits_{n\to\infty} \frac{3n^2-4n+1}{n-2n^2+9}$.

Compiling a small code in Python, it is possible to demonstrate stable dynamics of behaviour already for the first twenty elements of the numerical sequence $\frac{3n^2-4n+1}{n-2n^2+9}$. From fig. 1 the student can determine the limit value of the sequence.

Example 2. When studying monotonic sequences and the number e, it is possible to demonstrate to students the calculation of the number $e = \lim\limits_{n\to\infty}\left(1+\frac{1}{n}\right)^n$ with good accuracy by compiling the code in Python in several lines (fig. 2).

When studying the limit of a function at a fixed point, continuity and uniform continuity of a function, the teacher can also demonstrate these concepts in pictures. For example, Python tools (and functions of the *matplotlib.pyplot* library) can visually show the convergence of points on the graph to the limit value of the function.

Example 3. Draw a limit graphically: $\lim\limits_{x\to-1}\frac{x+1}{\sqrt{6x^2+3x+3}}$.

To solve the problem in the unit neighbourhood of the point $x=-1$, we consider a sequence of points that converge to their limit value -1 and calculate the value of the function $f(x)$ at these points. We mark the obtained coordinates (red dots in the fig. 3) on the function graph (blue colour).

In addition, students can be offered various tasks for evaluating limits of numerical sequences and the limits of functions at a fixed point using Python (for this, the appropriate functions of the *SymPy* library) and Maxima are used.

Derivative Visualization in Maxima. The study of differential calculus should begin with the visualization of the geometric interpretation of the derivative. Let us consider the graph of a function $y=f(x)$ at the neighbourhood of point x_0. At the indefinite approach of point $P(x_0+\Delta x;y_0+\Delta y)$ along the graph of a function $y=f(x)$ to the point $P_0(x_0;y_0)$, the secant line P_0P approaches some limiting position, which is called a tangent to the curve $f(x)$ at the point P_0 (Denisiuk et al. 2009).

Example 4. Let us demonstrate the geometric content of the derivative for the function $y=x^2-4x+10$ at the point $x_0=4$.

To solve this problem, we construct a family of secants and tangents at a point to the graph of this function at the point $x_0=4$. To find the equations of the secant and the tangent to the graph of the function $y=x^2-4x+10$ at the point x_0 and plot the graphs, let us use the CAS Maxima.

First, let us construct a equation of the tangent line of the form $y=kx+b$. The slope k of the tangent line of $f(x)=x^2-4x+10$ is given by taking the limit as Δx goes to 0 for the point $x_0=4$:

$$k=f'(x_0)=\lim_{\Delta x\to 0}\frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}=4.$$

Next, substituting the found value $k=4$ into the equation of the tangent line, we get the equation $y=4x-6$. Implementing this procedure in Maxima using the command *limit (expr, x, val)* to find the limit of the expression *expr* as the real variable x approaches the value *val* (Salz 2022), we obtain the dialog from fig. 4:

```
    →   The  slope   of  the tangent line
    →   k:=
(%i12) limit((f(4+delta)-f(4))/delta, delta, 0);
(%o12) 4
(%i13) k:4;
(%o13) 4
    →   The equation of  the tangent line
(%i14) yt:k·x0+b;
(%o14) b+16
(%i15) solve([yt=f(4)], [b]);
(%o15) [b=-6]
    →   the tangent line
(%i16) t(x):=4·x-6;
```

Figure 4

Let us now construct the family of secant lines for the graph of the function $y=x^2-4x+10$, which pass through points $(x_0; y_0)$ and $(x_1 = 4x + h; y_1)$, by using the equation $y=m(x-x_0)+y_0$, where $m = \frac{y_1-y_0}{x_1-x_0}$ is the slope. In the Maxima program, using the *ratsimp(expr)* command to the expression *expr* (Salz 2022), we get the general equation for this family of secant lines as shown on fig. 5.

We see that, as the values of $(x_1; y_1)= (4 + h; y(4 + h))$approach $(x_0; y_0) = (4; 10)$, the secant lines themselves approach the tangent line to the function $y=x^2-4x+10$ at x_0, which represents the limit of the secant lines. As the values of $h = 1.5, 0.5, 0.05$ get closer to 0, the secant lines also approach the tangent line $y=4x-6$ (red line in fig. 6). The graphical representation was obtained using the command *wxplot2d(explicit(f(x),x,xmin,xmax))* (Salz 2022).

```
Secant Lines
x1:4+h;
(%o17)  h+4
f(x1);
                    2
(%o18)  (h+4)  -4 (h+4)+10
The slope
m=(f(x1)-f(x0))/(x1-x0);
                   2
          (h+4)  -4 (h+4)
(%o19)  m=---------------
                  h
ratsimp(%);
(%o20)  m=h+4
The  equation of  the secant line
m:h+4;
(%o21)  h+4
y-f(x0)=m*(x-x0);
(%o25)  y-10=(h+4) (x-4)
ratsimp(%);
(%o26)  y-10=(h+4) x-4 h-16
The general equation of  the secant
s(x):=(h+4)*x-4*h-16+10;
```

Figure 5

```
(%i33) wxplot2d([x^2-4*x+10,4*x-6,5.5*x-12.0,4.5*x-8.0,4.05*x-6.2], [x,3,6])$
```

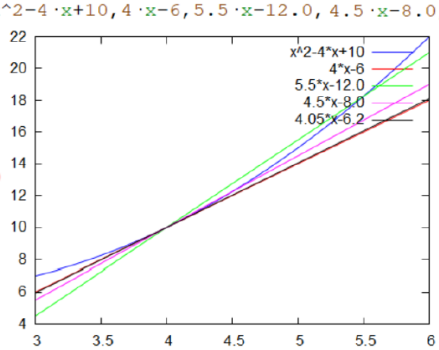

Figure 6

Applications of Derivatives in Python. The Taylor and Maclaurin formulas and later the Taylor/Maclaurin series are studied in the section on applications of the derivative, particularly for approximate calculations. For students to better un-

derstand the possibility of replacing some functions with a n-th degree polynomial (n is a nonnegative integer), the figure can demonstrate the gradual polynomial approximation of corresponding functions in the neighbourhood of zero.

Example 5. Let us demonstrate the graphical approximation of the Maclaurin series to the function $\cos(3x)$ in the neighbourhood of zero. Namely, by using the constructed graph, we verify the equality:

$$\cos(3x) = 1 - \frac{(3x)^2}{2!} + \frac{(3x)^4}{4!} - \cdots + (-1)^m \frac{(3x)^{2m}}{(2m)!} + \cdots \qquad (1)$$

We gradually increase the number of terms on the right-hand side to solve the problem and construct corresponding graphs of polynomial functions. It can be seen that when the number of terms in the polynomial increases, the partial sums on the right of (1) converge to the left-hand side near 0.

We get the result shown on fig. 7.

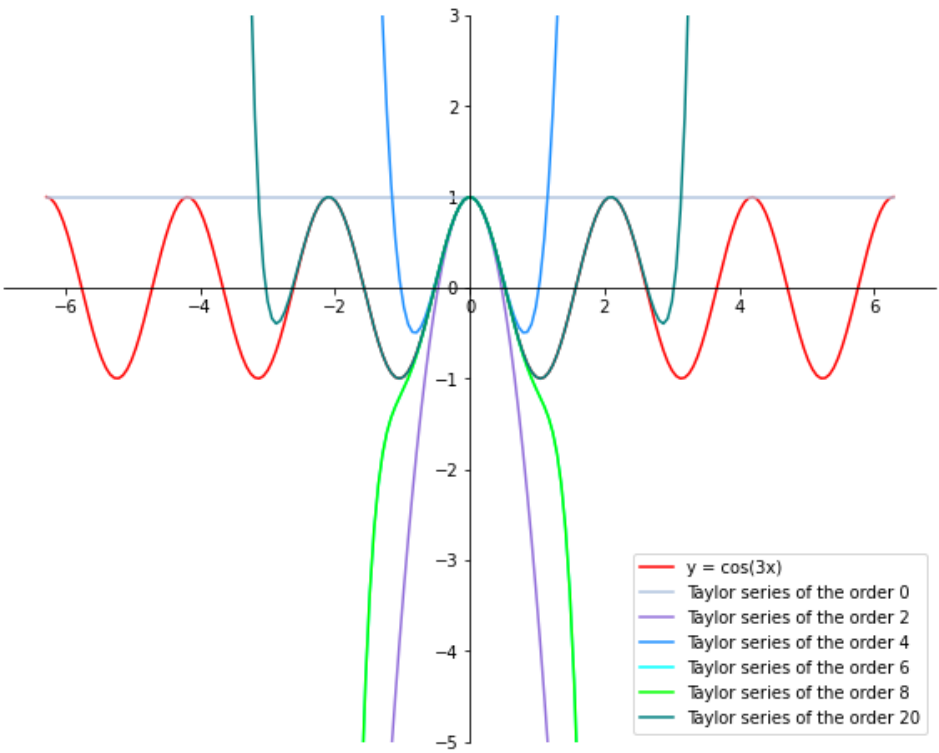

Figure 7

This figure was obtained by compiling a Python program using the *matplotlib.pyplot* library.

Applying Maxima for Integration. When studying integral calculus for students of IT programs, along with analytical calculations, it is worth showing methods of calculating integrals using computer algebra systems.

Using the Maxima system, it is possible to compute indefinite integrals and find antiderivatives using symbolic representation. In general, *integrate(f(x),x)* is a command that most users can easily use to compute various basic integrals $\int f(x)dx$ (Salz 2022). In Maxima, it is possible to find indefinite integrals from various types of integrands using only this command, in particular from the simplest irrational expressions and differential binomials. However, when integrating some irrationalities, for example, integrals of the form

$$\int R(x,\sqrt{ax^2+bx+c})dx \text{ and } \int R(x,X^{r_1/s_1},\cdots,X^{r_n/s_n})dx,$$

where R(•) is a rational function, $X=(ax+b)/(cx+d)$,r_i/s_i,$i\geq 1$ are fractions, the Maxima system does not immediately provide an answer. In such cases, it is necessary to use a standard integration method, such as the substitution technique.

Example 6. Find $\int \frac{dx}{\sqrt{(x-1)(x-2)^3}}$.

The Maxima system does not show the antiderivative when using the *integrate* command directly (fig. 8). Instead, it converts the entry from the previous command to the specified indefinite integral (fig. 9).

```
(%i27) integrate(1/sqrt((x-1)·(x-2)^3),x);
```

$$(\%o27)\quad \int \frac{1}{\sqrt{(x-2)^3\,(x-1)}}$$

Figure 8

However, after simplifying the integral function using the *radcan(expr)* command, where *expr* is an expression containing logs, exponentials, and radicals, we got an integral that can already be computed using the *integrate* command. The % symbol indicates that the *radcan* command is applied to the last line.; *nouns* causes the evaluation of noun forms (typically unevaluated functions such as *integrate* or *diff*) in *expr*.

Example 7. Find $\int \frac{dx}{2x+\sqrt[3]{(x-1)x^2}}$.

This integral is the integral of an irrational function and is found analytically by changing the variable. As can be seen, the direct application of the *integrate* command, even with the *radcan* simplification command, does not show the original (fig. 10). To solve this problem, we need to use the substitution rule in Maxima.

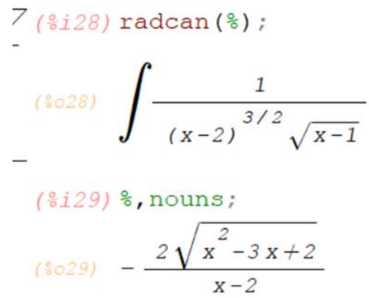

Figure 9

A function *changevar (expr, G (x, t), t, x)* (Maxima 5.46.0. Manual 2022) makes the change of variable in a given noun form integrate expression such that the old variable of integration is x, the new variable of integration is t, and x and t are related by the equation $G(x,t)$=0.

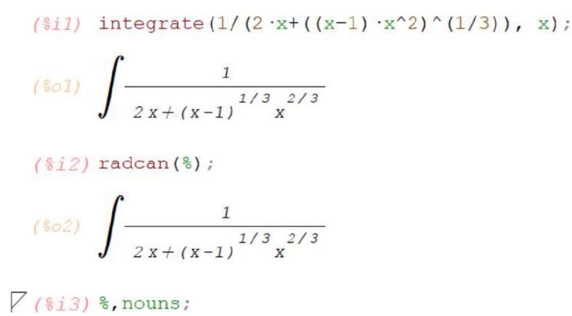

Figure 10

We use the substitution $(x-1)/x=t^3$, revealing the irrationality using the *changevar* command since it is impossible to express x in terms of t with the help of Maxima if there is a radical expression (fig. 11). As a result of this substitution, the CAS shows a new integral of the variable t in a non-calculable form. To compute such an integral, we need to enter the *%, nouns* command from the keyboard.

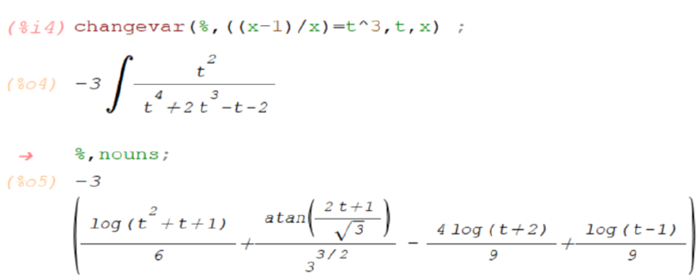

Figure 11

Also, in the Maxima system, many additional packages are designed for solving various mathematics problems. In particular, among them is the package *bypart. mac*, which contains the command of /integration by parts for an indefinite integral, that is, using the formula $\int f(x)dx=\int udv=uv-\int vdu$ (Strang G.1991). This command has the form *byparts(f(x),x,u(x),v'(x))* (Maxima 5.46.0. Manual 2022); as a result, the antiderivative of function $f(x)$ is displayed.

Example 8. Find $\int x^3 \ln x dx$.

To use the *bypart.mac* package, we need to preload it using the *load(bypart)* command. Next, in the description of the *byparts* command, we specify the integrand f, the variable of integration x, the functions $u=\ln x$ (we choose a function between $x^3$and $\ln x$, which is easier to differentiate than to integrate) and $v'(x)=x^3$. After the Shift+Enter combination, we obtain the result of integration by parts (fig. 12).

```
→  byparts(x^3·log(x),x,log(x),x^3);
(%o18)  x^4 log(x)/4 − x^4/16
```

Figure 12

The application of Maxima is also convenient when studying the topic „Integrating of rational functions". When integrating rational expressions of the form $\int P(x)/Q(x)dx$, where $P(x)$ and $Q(x)$ are polynomials, the Maxima system also uses the *integrate* command. However, it is often advisable to perform elementary transformations using the functions built into the package before using it in an integrand. For example, the *partfrac(f(x), x)* command expands the expression *f(x)* in partial fractions with respect to the main variable *x.*

Example 9. Find $\int x^5/(x^3-1)dx$.

After using the *partfrac* command to decompose the integral $x^5/(x^3-1)$, let us integrate the resulting expression using the main *integrate* command (fig. 13).

```
(%i24) partfrac(x^5/(x^3-1),x);
(%o24)  (2x+1)/(3(x^2+x+1)) + x^2 + 1/(3(x-1))

(%i26) integrate((2·x+1)/(3·(x^2+x+1))+x^2+1/(3·(x-1)),x);
(%o26)  log(x^2+x+1)/3 + x^3/3 + log(x-1)/3
```

Figure 13

Finding the Indefinite Integral in Python. In order to computethe indefinite integral in symbolic form, the *SymPy* library in Python is used (Vanderplas & Schanafelt 2017). SymPy is a powerful Python symbolic mathematics library that can be considered a full-featured computer algebra system. SymPy has various functions in symbolic computation, calculus, statistics, algebra, discrete mathematics, combinatorics, and physics. In particular, SymPy contains methods for computing indefinite, definite, and improper integrals. The *integrate()* command is used for such tasks. Let us give several ways to computeindefinite integrals.

```
In [10]: import sympy as sp
         x=sp.symbols('x')
         sp.Integral(x**3-2*x**2+12, x)
Out[10]: ∫ (x³ − 2x² + 12) dx

In [11]: fun=x**3-2*x**2+12
         sp.integrate(fun,x)
Out[11]: x⁴/4 − 2x³/3 + 12x

In [12]: import sympy as sp
         x=sp.symbols('x')
         int=sp.Integral(x**3-2*x**2+12, x)
         int.doit()
Out[12]: x⁴/4 − 2x³/3 + 12x
```

Figure 14

Example 10. Find $\int(x^3-2x^2+12)dx$.

The process of finding such an integral in Python is as shown on fig. 14.

Example 11. Find $\int \frac{x^5}{x^3-1}dx$.

Similarly, we get in Python as shown on fig. 15.

```
In [13]: fun2=x**5/(x**3-1)
         sp.integrate(fun2,x)
Out[13]: x³/3 + log(x³ − 1)/3
```

Figure 15

However, the *integrate ()* command does not integrate all functions. Similar to Maxima, integrals to which substitution rule or integration by parts is to be applied must be prepared first and then proposed the substitution. In such cases, models:

```
(%i2) integrate(1/(cos(x))^2,x,%pi/6,%pi/4);
(%o2) 1 − 1/√3
```

Figure 16

sympy.sympify(),sympy.integrals.transforms() are used.

Calculating Definite Integrals with Maxima. To evaluate definite integrals $\int_a^b f(x)dx$ in the Maxima package, use the *integrate*(*f*(*x*), *x, a, b*) command.

Example 12. Evaluate the integral $\int_{\pi/6}^{\pi/4} \frac{dx}{\cos^2 x}$.

Let us computethe integral using the *integrate* command (fig. 16).

As described above, in the Maxima system, the *changevar* command is used for the substitution method in integration. When calculating definite integrals, it is also used for specific substitutions. First, the integral is displayed on the screen with a new variable t and new limits of integration, and then it is computed using the command *%, nouns*. As with finding indefinite integrals, it is helpful to use this formula to check the efficiency of using different substitutions and when it is not possible to find the integral directly using the *integrate* command.

Example 13. Evaluate the integral $\int_0^4 \frac{1}{1+\sqrt{x}} dx$.

After entering the integral with the *integrate* command, apply $\sqrt{x} = t$ substitution using the *changevar(%,sqrt(x)=t,t,x)* command (fig. 17). An apostrophe before a Maxima command indicates a gap in the integral calculation. Next, the system questions whether the new variable t is negative or positive. To answer this question, enter the first letter p, which indicates positive values of the variable t. The following appearing line already contains an integral with a new variable t, and new limits of integration (found automatically). The last step is to use the *%, nouns* command to compute the result.

```
(%i17) 'integrate(1/(1+sqrt(x)),x,0,4);

(%o17)  ∫_0^4 1/(sqrt(x)+1)

(%i18) changevar(%,sqrt(x)=t,t,x);
Is t positive, negative or zero?p;

(%o18)  2 ∫_0^2 t/(t+1)

(%i19) %,nouns;
(%o19)  2 (2-log(3))
```

Figure 17

Evaluating Definite Integrals in Python. The SymPy library also is used to compute the definite integral in Python. However, since not all indefinite integrals can be computed in the symbolic form, and therefore not all definite integrals can be analytically computed, it is necessary to use the methods of their approximate calculation. For IT students, it would be interesting not only to compute integrals approximately but also to evaluate the accuracy of these approximations and to conclude which method gives more accurate results. For this purpose, it is possible to take such an integral, which is easily calculated analytically, and compute it on Python with various functions, then compare the results and conclude the method‘s accuracy.

Let us consider the most popular methods of numerical integration (Epperson 2013).

1. Right Riemann sums:

$\int_a^b f(x)dx \approx h\sum_{k=0}^{n-1} f(x_k)$, where $h = \frac{b-a}{n}, x_k = a + kh$

with a calculation error $\Delta \leq \frac{M_1(b-a)^2}{2n}$, where $|f'(x)| \leq M_1$.

An alternative can be the formula for median rectangles:

$$\int_a^b f(x)dx \approx h\sum_{k=0}^{n-1} f(x_k + \frac{h}{2}), \quad (2)$$

with an error $\Delta \leq \frac{M_2(b-a)^3}{24n^2}$, where $|f''(x)| \leq M_2$. This formula is generally more accurate.

2. The trapezoidal rule:

$$\int_a^b f(x)dx \approx h(\frac{f(a)+f(b)}{2} + \sum_{k=1}^{n-1} f(x_k)) \quad (3)$$

with an error $\Delta \leq \frac{M_2(b-a)^3}{12n^2}$, where $|f''(x)| \leq M_2$.

3. Simpson's rule:

$$\int_a^b f(x)dx \approx \frac{h}{3}(f(a) + f(b) + 2\sum_{k=1}^{n-1} f(x_{2k}) + 4\sum_{k=0}^{n-1} f(x_{2k+1})), \quad (4)$$

where $h = \frac{b-a}{2n}, x_k = a + kh$, with an error $\Delta \leq \frac{M_3(b-a)^5}{180n^4}$, where $|f^{(4)}(x)| \leq M_3$.

Let us implement these methods in Python.

Example 14. Evaluate the integral $\int_0^{\sqrt{\pi/2}} x\sin(x^2)\,dx$.

```
1]: from scipy.integrate import quad
    import numpy as np
    func = lambda x: x*np.sin(x**2)
    int = quad(func, 0, np.sqrt(0.5*np.pi))
    print('int=', int)

    (0.4999999999999998, 5.55111512312578e-15)
```

Figure 18

1) If we analytically calculate the given integral using the substitution rule, we get result

$$\int_0^{\sqrt{\pi/2}} x\sin(x^2)\,dx = 0.5. \quad (5)$$

2) Let us apply the Midpoint Rule in Python.

a) We use the ready-made *quad* function from the *scipy.integrate* package (fig. 18). It gives the result of integration and the accuracy of calculations.

b) Then we program formula (2) of the midpoint rule (fig. 19).

3) Let us use the trapezoidal rule (fig. 20).

a) If we compare with the result (5), we can see that the trapezoidal rule gives worse accuracy than the midpoint rule.

b) Then we program formula (3) of the trapezoidal rule (fig. 21).

We can see that the formula we programmed gives a better result than the ready-made *trapz* function.

4) Let us apply Simpson's rule that is, formula (4) as shown on fig. 22. This method also gives a good result.

Therefore, comparing the analytically computed integral (5), we can conclude that the results obtained by the Median rectangles formula and Simpson's rule are more accurate.

```
In [2]: def int_avr(f, a, b, n):
            h = (b - a)/n
            xi = [(a + (i + 1/2) * h) for i in range(n)]
            f_i = [f(x) for x in xi]
            return h * sum(f_i)
        func = lambda x: x*np.sin(x**2)
        int_avr(func, 0, np.sqrt(0.5*np.pi), 100)

Out[2]: 0.499993453704887
        By increasing the number of nodes, we get a more accurate result

In [3]: int_avr(func, 0, np.sqrt(0.5*np.pi), 300)

Out[3]: 0.499999272763301
```

Figure 19

```
In [6]: from scipy.integrate import trapz
        x=np.arange(0, np.sqrt(0.5*np.pi),0.005)
        y=func(x)
        int2=trapz(y,x=x)
        print('int2=', int2)

        int2= 0.49585402149948926
```

Figure 20

```
In [7]:
def int_trpz(f, a, b, n):
    h = (b - a)/n
    xi = [(a + i * h) for i in range(1, n)]
    f_i = [f(x) for x in xi]
    return h *((f(a)+f(b))*0.5 + sum(f_i))
int_trpz(func, 0, np.sqrt(0.5*np.pi), 100)

Out[7]: 0.5000130914670029
```

Figure 21

```
In [1]: import math
        import numpy as np
        func=lambda x: x*np.sin(x**2)
        def int_simps(f,a,b,n):
          h=(b-a)/(2*n)
          x2i=[(a+i*h) for i in range(2,2*n,2)]
          x2i_1=[(a+i*h) for i in range(1,2*n+1,2)]
          f_2i=[f(x) for x in x2i]
          f_2i_1=[f(x) for x in x2i_1]
          return h/3*(f(a)+f(b)+2*sum(f_2i)+4*sum(f_2i_1))
        int_simps(func,0,np.sqrt(0.5*np.pi),100)

Out[1]: 0.49999999962559233
```

Figure 22

Conclusion

As we can see, a problem in calculus can be visualized using CAS tools or a programming language. There are two main aspects of CAS usage. The first aspect is direct visualization of the appropriate way of solving a particular problem for a better understanding of the use of the method to show its practical application. The second aspect is a significant saving of calculation time.

Depending on the numerical or symbolic task, the choice arises to choose the appropriate type of CAS. This is exactly what the teacher's role should be, that is, to guide the student in choosing the software that is most efficient and quickly

able to solve the task. In the Maxima system, most mathematical analysis methods are already programmed. The user only needs to learn the appropriate syntax of the functions and use them properly. However, without understanding the mathematical methods used, it is not always possible to apply them correctly. There are several tasks where the user must first perform specific actions (transformation, simplification) to use the syntax of the corresponding Maxima function.

It is worth showing the capabilities of Python during learning of various fundamental disciplines from the very first year for students of IT specialties to gain programming skills and experience. In addition, thereby continue to develop the ability of students to program the formulas, methods or algorithms necessary to solve many mathematical problems.

Therefore, this article shows the capabilities of Maxima and Python and, accordingly, gives the student a choice for the applied application of the theory of calculus.

NOTES

1. MAXIMA. A Computer Algebra System, [viewed 13 January 2023]. Available from: https://maxima.sourceforge.io/index.html.
2. MAXIMA 5.46.0. Manual [viewed 13 January 2023]. Available from: https://maxima.sourceforge.io/docs/manual/maxima_toc.html#SEC_Contents

REFERENCES

DE SOUZA, P.N., et al., 2004. *The Maxima Book,* [viewed 13 January 2023]. Available from: https://maxima.sourceforge.io/docs/maximabook/maximabook-19-Sept-2004.pdf.

DENISIUK, V. P., et al., 2009. *Higher mathematics*. Part 1: Manual. Kyiv: NAU.

EPPERSON, J. F., 2013. *An Introduction to Numerical Methods and Analysis*, 2rd edition. Wiley.

KADO, K., 2022. A teaching and learning the fundamental of calculus through Python-based coding. *International Journal of Didactical Studies,* vol. 3, no. 1, 15006 [viewed 13 January 2023]. Available from: https://doi.org/10.33902/IJODS.202215006.

KARJANTO, N., 2021. Calculus and Digital Natives in Rendezvous: wxMaxima Impact. *Educ. Sci.*, no. 11, pp. 490, [viewed 13 January 2023]. Available from: https://doi.org/10.3390/educsci11090490.

KARJANTO, N., HUSAIN, H.S., 2021. Not Another Computer Algebra System: Highlighting wxMaxima in Calculus. *Mathematics*, no. 9, pp. 1317 [viewed 13 January 2023]. Available from: https://doi.org/10.3390/math9121317.

MOHD AYUB, A.F. et al., 2012. WxMaxima Computer Software as an Aid to the Study of Calculus by Students with Different Learning Approaches, *Procedia – Social and Behavioral Sciences*, no. 64, pp. 467 – 473, [viewed 13 January 2023]. Available from: https://doi.org/10.1016/j.sbspro.2012.11.055.

PEREIRA JUNIOR, R.A., et al., 2022. Cálculo diferencial e integral evoluindo a inteligência computacional Mediante a linguagem Python de programação[in Portuguese][Differential and integral calculus evolving computational intelligence through the Python programming language]. *Brazilian Journal of Development*, vol. 8, no. 9, pp. 64737 – 64761 [viewed 13 January 2023]. Available from: https://doi.org/10.33902/IJODS.202215006.

PARK, K.-E., et al., 2019. Teaching and Learning of University Calculus with Python-based Coding Education. *Communications of Mathematical Education*, vol. 33, no. 3, pp. 163 – 180 [viewed 13 January 2023]. Available from: https://doi.org/10.7468/JKSMEE.2019.33.3.163.

SALZ, S., 2022. *CAS Maxima Workbook.* [viewed 13 January 2023]. Available from: https://roland-salz.de/Maxima_Workbook.pdf.

STRANG, G., 1991. *Calculus*. Wellesley-Cambridge Press.

VANDERPLAS, J., SCHANAFELT, D., 2017. *Python Data Science Handbook: Essential Tools for Working With Data*. 1a ed. Sebastopol, CA: O'Reilly Media.

✉ **Dr. Oleksandr Pohoriliak, Assoc. Prof.**
ORCID iD: 0000-0002-0501-4861
Department of Probability Theory and Mathematical Analysis
Uzhhorod National University
3, Narodna Square
88000 Uzhhorod, Ukraine
E -mail: oleksandr.pohoriliak@uzhnu.edu.ua

✉ **Dr. Olga Syniavska, Assoc. Prof.**
ORCID iD: 0000-0002-2711-3940
Department of Probability Theory and Mathematical Analysis
Uzhhorod National University
3, Narodna Square
88000 Uzhhorod, Ukraine
E-mail: olga.synyavska@uzhnu.edu.ua

✉ **Dr. Anna Slyvka-Tylyshchak, Assoc. Prof.**
ORCID iD: 0000-0002-7129-0530
Department of Probability Theory and Mathematical Analysis
Uzhhorod National University
3, Narodna Square
88000 Uzhhorod, Ukraine
E-mail: anna.slyvka@uzhnu.edu.ua

✉ **Dr. Antonina Tegza, Assoc. Prof.**
ORCID iD: 0000-0001-5310-4311
Department of Probability Theory and Mathematical Analysis
Uzhhorod National University
3, Narodna Square
88000 Uzhhorod, Ukraine
E-mail: antonina.tegza@uzhnu.edu.ua

✉ **Prof. Dr. Alexander Tylyshchak**
ORCID iD: 0000-0001-7828-3416
Department of Mathematics and Informatics
Ferenc Rákóczi II Transcarpathian Hungarian Institute
6, Kossuth square
90202 Berehove, Ukraine
E-mail: alxtlk@gmail.com