UDC 510.2; 621.3

DOI: 10.20535/2523-4455.mea.328241

Software Support for the Higher Mathematics Course at the Technical University

Part 2. Practical examples of basic function analysis

O. V. Bogdanov, Assoc. Prof., PhD, 0000-0002-0911-5563
Yu. P. Butsenko, Assoc. Prof., PhD, 0000-0003-4806-9587
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" 00syn5v21
Kyiv, Ukraine

O. I. Balina, Assoc. Prof., PhD, 0000-0001-6925-0794
I. S. Bezklubenko, Assoc. Prof., PhD, 0000-0002-9149-4178
Kyiv National University of Construction and Architecture 02qp15436
Kyiv, Ukraine

Abstract—This paper explores the integration of Scilab software into the higher mathematics curriculum for engineering students, focusing on enhancing the understanding and application of function analysis. It addresses the necessity of equipping future engineers with skills in both theoretical mathematics and computational tools. The study provides practical examples, such as plotting functions, finding asymptotes, and interpolation techniques, demonstrating how Scilab can aid in solving complex mathematical problems. The authors argue for a balanced educational approach, combining traditional mathematical methods with the effective use of software to enhance learning outcomes in engineering education.

Keywords — Educational technology; Hybrid learning; Electronic learning; Engineering education; Electrical engineering; Calculus; Mathematical programming.

I. INTRODUCTION

In the first part [1] the basic problems of modern mathematical training of engineers in the specialty G5 Electronics, electronic communications, instrument making, and radio engineering were considered. The problem cannot be fully disclosed within the framework of one work, and we will continue evaluating the use of mathematical software packages, using the example of Scilab, in the educational process.

In the previous work, the use of the Scilab program was justified; however, it should be noted that this software is constantly updated, so now we will use the new version of Scilab 2025.0.0 for examples. [2].

The features of this package make it convenient, first of all, for getting acquainted with the tasks and methods of computational mathematics [3], while the first (and extremely important!) task is to convey to students an understanding of at least the practical use of mathematical formulas known to them (according to O. M. Krylov), set out in [4]: firstly, in applied problems there is simply no need to use accurate formulas, secondly, there is no need to achieve absolute accuracy in calculations, thirdly, it is possible to use even certainly inaccurate formulas, while controlling the compliance of the obtained result with the requirements imposed on it, and, finally, the fact that the user of computational methods is not at all interested in the process of calculations, but only their result, which must be achieved with the least expenditure of time and effort, but with acceptable accuracy. Students should be reminded of such key concepts as absolute and relative errors, examples of their evaluation, and accumulation. At the same time, it is necessary to pay due attention to familiarizing and accustoming students to the discrete specification of functional dependencies, that is, considering them as an alternative to the continuous approach characteristic of classical differential and integral calculus, discrete templates characteristic of applied problems, and computer calculations.

In this sense, it seems appropriate to consider tasks such as the construction of discrete graphic templates for an explicitly given function with the identification of its vertical and oblique asymptotes, a function given implicitly (in this case, the issue of approximate finding of the roots of equations is also considered), the construction ‘by points’ of a line given by an equation in a polar coordinate system on a plane.

When designing electronic devices, it is necessary to be able to analyze electrical signals in a numerical representation that is presented before the physical modeling of the device. This need arises for engineers not only when designing new systems, but also when analyzing the operation of existing circuits to identify the causes of errors or mistakes.

DOI: 10.20535/2523-4455.mea.328241

Any numerical analysis is based on the ability to obtain information about a function using mathematical methods. Therefore, let us consider examples of combining classical analytical research with "pen and paper" and the capabilities that Scilab provides for engineers.

II. PRACTICAL PLOTTING EXAMPLES

A. Plotting a function graph by points, finding asymptotes

As an example, consider the function:

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

We will not provide the textual part of the classical function analysis that our students are familiar with, but instead we will focus on scripts that can become the basis for building a new educational process.

The asymptotes plot (Fig. 1) for the function was constructed using the script:

```
// First graph: vertical asymptote

x1 = linspace(-2, -1.02, 100); // Left side

x2 = linspace(-0.98, 0, 100); // Right side

// Define the function

f1 = (x1.^2 - x1 + 1) ./ (x1 + 1);

f2 = (x2.^2 - x2 + 1) ./ (x2 + 1);

// Second graph: on the interval from 0 to 10, with an oblique asymptote

x3 = linspace(0, 10, 50); // New interval for the second graph

// Define the function for the new range

f3 = (x3.^2 - x3 + 1) ./ (x3 + 1);

figure(1); // Separate window for the first graph

clf; // Clear the plot

subplot(1, 2, 1);

// Plot the function by points

plot(x1, f1, 'bo', 'MarkerSize', 2); // Left side of the function with points

plot(x2, f2, 'bo', 'MarkerSize', 2); // Right side of the function with points

// Add a vertical asymptote at x = -1

x_asymptote = [-1, -1];

y_asymptote = [-160, 160]; // Range for the vertical asymptote

plot(x_asymptote, y_asymptote, 'r--', 'LineWidth', 1.5); // Red dashed line

// Add labels to the plot

xlabel("x");

ylabel("f(x)");

title("Graph of the function f(x) = (x^2 - x + 1) / (x + 1)");

xgrid; legend(["f(x)", "Vertical Asymptote"]);

subplot(1, 2, 2);

// Plotting a function by points

plot(x3, f3, 'bo', 'MarkerSize', 2); // Function for a new interval

// Add the slope of the asymptote y = x - 2

x_range = linspace(0, 10, 50);

y_asymptote_oblique = x_range - 2;

plot(x_range, y_asymptote_oblique, 'g--', 'LineWidth', 1.5); // Green dashed line
```

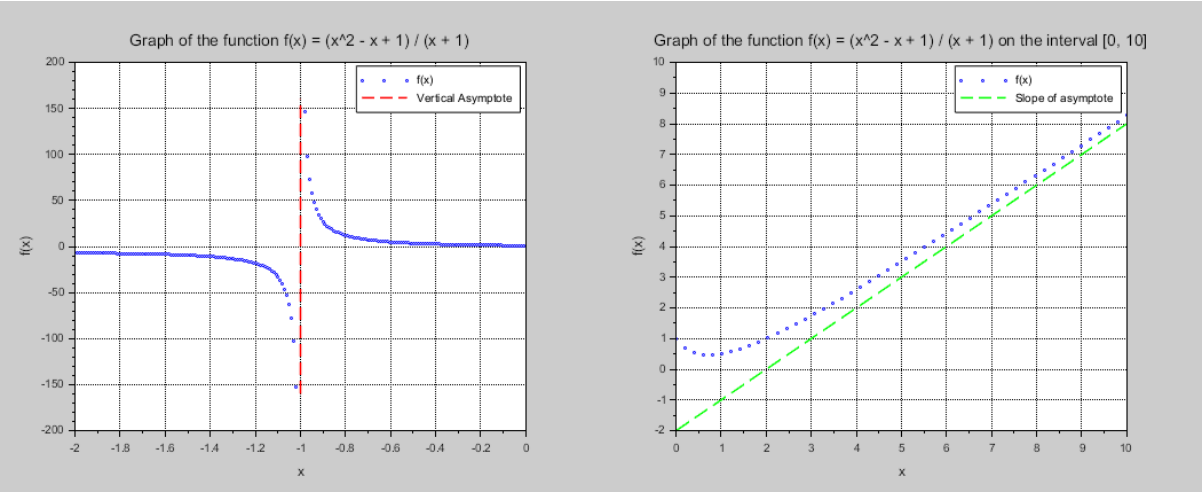

Fig. 1 Asymptote plotting

DOI: 10.20535/2523-4455.mea.328241

```
// Add labels to the graph
xlabel("x");
ylabel("f(x)");
title("Graph of the function f(x) = (x^2 - x + 1) / (x + 1) on the interval [0, 10]");
xgrid; legend(["f(x)", "Slope of asymptote"]);
```

B. Construct a line given in a polar coordinate system through points

Consider a function as an example:

$$\rho = 2 \cdot \sin(4\varphi) \qquad (2)$$

The polar plots (Fig. 2) for the function (2) were constructed using the script:

```
// Definition of tabular angles (in degrees)
angles_deg = [0, 30, 45, 60, 90, 120, 135, 150, 180, -30, -45, -60, -90, -120, -135, -150, -180];
// Addition of angles that become tabular after division by 8
additional_angles = angles_deg / 8;
angles_deg = unique([angles_deg, additional_angles]);
// Definition of radius vector (function ro = 2*sin(4*phi))
ro = 2 .* sin(4 .* angles_deg);
// Conversion from polar to Cartesian
x_dot = ro .* cos(angles_deg);
y_dot = ro .* sin(angles_deg);
// Determine the number of points for the second graph
n = 500;
// Create a vector of angles (in radians)
phi = linspace(-%pi, %pi, n); // From -π to π
// Define the radius vector (function r = 2*sin(4*phi))
ro = 2 .* sin(4 .* phi);
// Convert from polar to Cartesian
x = ro .* cos(phi);
y = ro .* sin(phi);
// Plot graphs in one window
figure(1); // Separate window for graphs
clf; // Clear the graph
// Left graph: Tabular angles
subplot(1, 2, 1); // Left part of the window
plot2d(x_dot, y_dot, style=-1, rect=[-2, -2, 2, 2]); // Plotting the graph
// Add labels
title("Polar plot: Tabular angles");
xlabel("X");
ylabel("Y");
// Add grid
gca().grid = [1, 1];
// Right plot: Solid plot
subplot(1, 2, 2); // Right part of the window
```

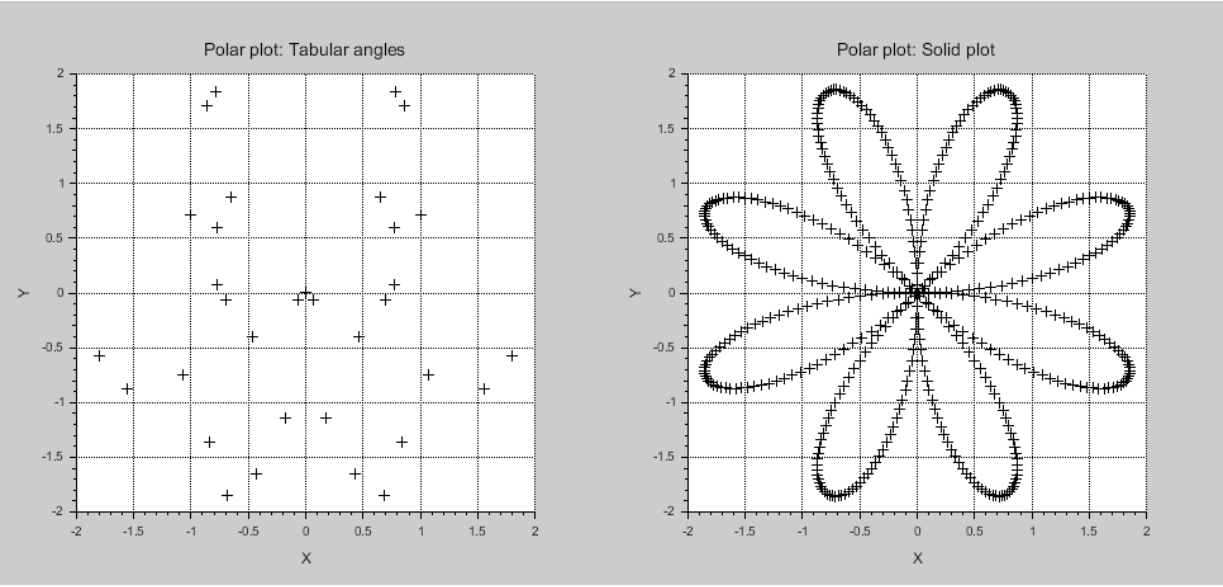

Fig. 2 Polar plotting

DOI: 10.20535/2523-4455.mea.328241

```
plot2d(x, y, style=-1); // Plotting the graph
// Add labels
title("Polar plot: Solid plot");
xlabel("X");
ylabel("Y");
// Add grid
gca().grid = [1, 1];
```

The convenience of this approach lies not only in the speed of obtaining results, which frees up class time for a more thorough analysis of the results, but also relieves students of the problem of selecting a basic scale for "manual" plotting of graphs.

It is also necessary to remember that not every engineering student has the 'graphic' skills to construct complex graphs, for example, in a polar coordinate system. The use of modern software tools will, in our opinion, increase the effectiveness of training electronics engineers.

III. PRACTICAL EXAMPLES OF "RESTORING" FUNCTIONS

Analyzing functions by their graphs is a fundamental skill; on the other hand, methods that allow one to "restore" the analytical specification of a function or its close analogue by a discrete template is fundamentally important, primarily polynomial interpolation.

One of the typical problems that requires the use of interpolation methods is the problem of synthesizing antenna arrays [3], [5]–[8]. At the same time, an important task is to convey to students the understanding that interpolation procedures in the vast majority of cases only give an idea of some analogue of the functional dependence that is actually observed. As for the possibilities of extrapolation, that is, predicting the values of the function outside the given interval, there may be fundamental deviations of the extrapolated values from the real ones, which can be illustrated by the example of interpolation of a somewhat complicated, compared to a polynomial, function and comparing its values with the values of the interpolation polynomial outside this interval.

A. Plotting the Lagrange interpolation polynomial

Let us consider the problem of constructing a Lagrange interpolation polynomial for the function:

$$\ln(1 + x) \tag{3}$$

in the range of values from 0 to 1 with a discreteness of 0.1, draw graphs and compare the behaviour of the function and the polynomial from 1 to 2.

The Lagrange interpolation (Fig. 3) for the function was constructed using the script:

```
// Determining points for the function ln(1+x) values from 0 to 1 with a discreteness of 0.1
x_vals = 0:0.1:1;
y_vals = log(1 + x_vals);
// constructing a Function for calculating the Lagrange polynomial
function L=lagrange_interpolation(x, x_vals, y_vals)
  L = 0;
  n = length(x_vals);
  for i = 1:n
    term = y_vals(i);
```

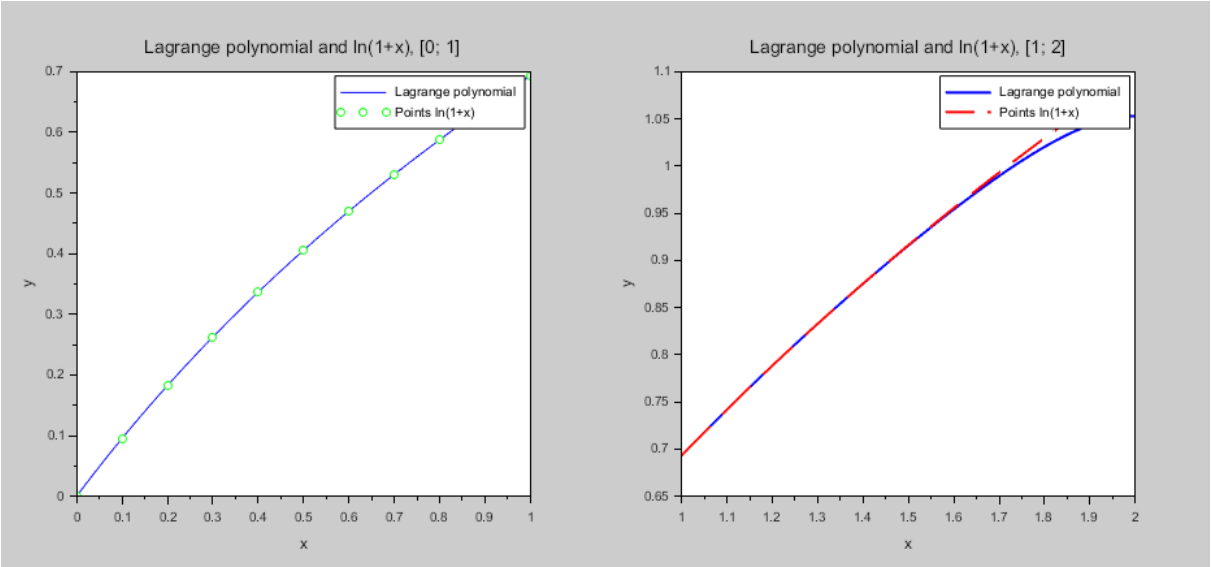

Fig. 3 Lagrange polynomial

DOI: 10.20535/2523-4455.mea.328241

```
        for j = 1:n
            if i <> j then
                term = term * (x - x_vals(j)) / (x_vals(i) - x_vals(j));
            end
        end
        L = L + term;
    end
    endfunction
// 1 graph
// Calculating values for the Lagrange polynomial for 1 graph
x_interpol1 = 0:0.01:1; // Interval for the first plot
y_interpol1 = zeros(1, length(x_interpol1));
for i = 1:length(x_interpol1)
    y_interpol1(i) = lagrange_interpolation(x_interpol1(i), x_vals, y_vals);
end
// Plotting the first plot
figure();
clf;
subplot(1, 2, 1)
plot(x_interpol1, y_interpol1, "b");
plot(x_vals, y_vals, "go");
xtitle("Lagrange polynomial and ln(1+x), [0; 1]", "x", "y");
legend("Lagrange polynomial", "Points ln(1+x)");
// 2nd plot
// Calculating values for the Lagrange polynomial for the 2nd plot
x_interpol2 = 1:0.01:2;
y_interpol2 = zeros(1, length(x_interpol2));
y_true2 = log(1 + x_interpol2);
for i = 1:length(x_interpol2)
    y_interpol2(i) = lagrange_interpolation(x_interpol2(i), x_vals, y_vals);
end
// Construction of the second graph
subplot(1, 2, 2)
plot(x_interpol2, y_interpol2, "b", "LineWidth", 2);
plot(x_interpol2, y_true2, "r--", "LineWidth", 2);
xtitle("Lagrange polynomial and ln(1+x), [1; 2]", "x", "y");
legend("Lagrange polynomial", "Points ln(1+x)");
```

B. Construction of extrapolation

In the case when measurement errors are observed when determining the functional dependence, which is practically inevitable in every real experiment, the methods of 'function type recognition" [3] are used first, followed by their approximation in one way or another, the simplest in this case being the least squares method.

Let us consider a function given in Table 1

Table 1 Numerical definition of a function

U	0.5	1	1.5	2	2.5	3	3.5	4	4.5
V	12	10.1	11.58	17.4	30.68	53.6	87.78	136.9	202.5

and show the way to restore it using the least squares method:

$$V = \frac{1}{A + Be^{-U}}. \quad (4)$$

The first step is to determine the constant coefficients A and B:

```
// Data
U = [0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5];
V = [12, 10.1, 11.58, 17.4, 30.68, 53.6, 87.78, 136.9, 202.5, 287];
// Modeling function V = 1 / (A + B * exp(-U))
function res=model(U, A, B)
    res = 1 / (A + B * exp(-U));
endfunction
// Minimization function S(A, B)
function S=cost_function(params)
    A = params(1);
    B = params(2);
    S = 0;
    for i = 1:length(U)
        S = S + (V(i) - model(U(i), A, B))^2; // Sum of squared errors
    end
```

DOI: 10.20535/2523-4455.mea.328241

```
endfunction

// Initial values for parameters A and B

initial_params = [1, 1]; // Example of initial values

// Minimizing a function using the least squares method

optimized_params = fminsearch(cost_function, initial_params);

// Output of results

disp("Optimized values of parameters A and B:");

disp(optimized_params);
```

This code outputs the least squares optimal parameters A and B to the Scilab console:

```
"Optimized values of parameters A and B:"
 0.0011836    0.3400271
```

Therefore, the function $V(U)$ has the form:

$$V = \frac{1}{0.0011836 + 0.3400271e^{-U}} \qquad (5)$$

Let us compare the plot (Fig. 4) of the function $V(U)$ and the polynomial obtained by the finite difference method (5), in the interval from -3 to 8 :

```
// Determine the range for U

U = linspace(-4, 8, 500);

// Calculate the value for the function V

V1 = 1 ./ (0.0011836 + 0.3400271 * exp(-U));

// Calculate the value for the polynomial

V2 = 15.044431 - 5.745276 * U - 1.5882351 * U.^2 + 2.3972827 * U.^3 + 0.065352 * U.^4;

// Create a graph

clf;

plot(U, V1, 'b-', 'LineWidth', 2);

plot(U, V2, 'r--', 'LineWidth', 2);

// Add labels

xlabel('U', 'FontSize', 3);
```

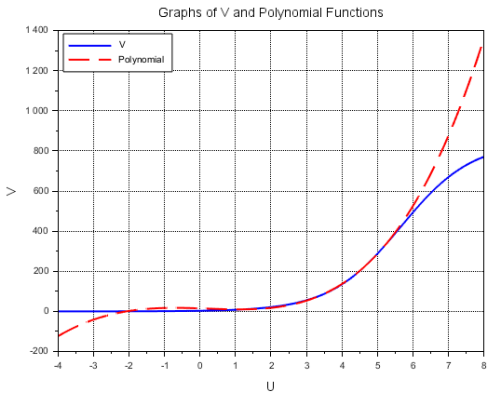

Fig. 4 Extrapolation results

```
ylabel('V', 'FontSize', 3);

legend(["V", "Polynomial"], 'FontSize', 2);

// Add a title

title('Graphs of V and Polynomial Functions', 'FontSize', 3);

xgrid(1); // Display the grid
```

The results presented will clearly show students the problem of the reliability of extrapolated data outside the given interval.

Conclusion

The present part provides practical examples of using the Scilab software package for the fundamental analysis of function properties. The issue of interpolation and extrapolation of experimental results, an integral professional task of an engineer, is separately considered.

As mentioned above, using Scilab in the educational process is not a full-fledged replacement for the traditional academic process but only a powerful auxiliary tool. This tool helps to reduce the time of standard constructions significantly and to increase the time for detailed analysis of the results obtained. You can also quickly change the input parameters of the functions, demonstrating the dynamics to students.

References

[1] O. V. Bogdanov, Y. P. Butsenko, O. I. Balina, and I. S. Bezklubenko, “Software Support for the Higher Mathematics Course at the Technical University,” *Microsystems, Electronics and Acoustics*, vol. 29, no. 2, Aug. 2024, **DOI:** 10.20535/2523-4455.mea.314217.

[2] “Scilab 2025.0.0 - Announcements - Scilab community.” [Online]. Available: https://scilab.discourse.group/t/scilab-2025-0-0/684. [Accessed: 22-Jan-2025].

[3] S. Rosloniec, *Fundamental Numerical Methods for Electrical Engineering*. Berlin: Springer-Verlag Berlin Heidelberg, 2008.

[4] O. I. Balina, I. S. Bezklubenko, and Y. P. Butsenko, *Numerical methods: lecture notes [Chysel’ni metody: konspekt lektsiy]*. Kyiv, Ukraine: KNUBA, 2019, **URL:** https://repositary.knuba.edu.ua/handle/123456789/13229.

[5] V. T. Hrinchenko, I. V. Vovk, and V. T. Matsypura, *Volnovyye zadachi akustiki [Wave problems of acoustics]*. Kyiv,Ukraine: Naukova dumka, 2007.

[6] O. I. Starovoit, O. H. Leiko, A. V. Derpa, and O. V. Bogdanov, *Obiemni systemy z tsylindrychnymy piezokeramichnymy vyprominyuvachamy i ekranom [Volumetric systems with cylindrical piezoceramic radiators and a screen]*. Kyiv,Ukraine: PH Burago D., 2022.

[7] I. V. Vovk and V. T. Matsypura, “Izlucheniye zvuka reshetkoy, obrazovannoy soosnymi tsilindricheskimi p’yezokeramicheskimi obolochkami s tortsevymi ekranami. Chast’ I. Teoriya [Sound radiation from a lattice formed by coaxial cylindrical piezoceramic shells with end screens. Part I. Theory],” *Akustychnyy visnyk*, vol. 4, no. 2, pp. 11–17, 2001.

[8] C. H. Sherman and J. L. Butler, *Transducers and Arrays for Underwater Sound*. New York, NY: Springer New York, 2007, **ISBN:** 978-0-387-32940-6.

Надійшла до редакції 27 січня 2025 року
Прийнята до друку 28 квітня 2025 року

DOI: 10.20535/2523-4455.mea.328241

УДК 510.2; 621.3

Програмні засоби супроводження курсу вищої математики у технічному університеті

Частина 2. Практичні приклади базового аналізу функцій

О. В. Богданов, доц., канд. техн. наук, 0000-0002-0911-5563
Ю. П. Буценко, доц., канд. техн. наук, 0000-0003-4806-9587
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського» 00syn5v21
Київ, Україна

О. І. Баліна, доц., канд. техн. наук, 0000-0001-6925-0794
І. С. Безклубенко, доц., канд. техн. наук, 0000-0002-9149-4178
Київський національний університет будівництва і архітектури 00syn5v21
Київ, Україна

Анотація—У статті розглядаються актуальні проблеми математичної підготовки сучасних інженерів, зокрема у контексті спеціальності G5 Електроніка, електронний зв'язок, приладобудування та радіотехніка.

Стаття є другою частиною і в ній розглядаються приклади використання Scilab для аналізу функцій, включаючи побудову графіків, знаходження асимптот, інтерполяцію та екстраполяцію. Особлива увага приділяється інтерполяції та екстраполяції експериментальних даних.

Автори підкреслюють, що Scilab є допоміжним інструментом, який дозволяє оптимізувати обчислення та аналіз результатів.

Ключові слова — Освітня технологія; Гібридне навчання; Електронне навчання; Інженерна освіта; Електротехніка; обчислення; Математичне програмування.