



On a computer program determining types of functional equations

ATTILA GILÁNYI AND LAN NHI TO

Dedicated to the 100th anniversary of the birth of János Aczél.

Abstract. In this paper a computer program, developed in the computer algebra system Maple is presented, which determines the type of functional equations.

Mathematics Subject Classification. Primary 39B52, 39B72, 68W30; Secondary 39A70.

Keywords. Functional equations, Systems of functional equations, Computer-assisted methods, Computational mathematics, Symbolic computation.

Introduction

The continuous development and improvement in computer technology and computational science provided solid support for mathematical models, making them more efficient in terms of computing time and resources. Computers help solve problems quickly and accurately, offering scientists effective ways of treating unsolved problems or topics that are yet to be studied in depth. Due to some basic features of the theory of functional equations, the application of computer-assisted methods to problems related to this field started relatively late, at the end of the 20th century.

In the 1980s, János Aczél posed the question if it was possible to develop a computer program determining the solutions of two variable linear functional equations using László Székelyhidi's method published in his paper [35]. The first program to addressing such equations was created 30 years ago and described in the thesis [22] (cf. also [23]; and, for more general versions, [8], [9]

This project was supported by the Scientific Council of the University of Nyíregyháza. Supported by the University of Debrecen Program for Scientific Publication.

and [10]). Over the last few decades, many researchers have used computers to study functional equations (cf., e.g., [4], [5], [6], [14], [20], [21], [24], [29], [30], [31], [33] and [32]).

Today, symbolic calculations combined with general computer programming are strong methods for exploring functional equations. We are currently developing a software package that includes programs related to functional equations. It comes as a natural idea to create a program capable of determining the types or classes functional equations belong to. In this paper we present this program and its applications.

In the first part of the paper, we list the types and classes of functional equations considered in our program. In the main section, we present our program with a full description as well as examples of the running program. In the last section, we show some short codes and detailed explanations to illustrate how our program works.

1. Types of functional equations considered

In this section, we shortly describe the functional equations recognized by our computer program.

The *Cauchy functional equation*

$$f(x + y) = f(x) + f(y) \quad (1)$$

is fundamental not only to the theory of functional equations but in all of mathematics. It is named after Augustin-Louis Cauchy, who was the first to determine its general continuous solutions [11]. It had been studied earlier by several authors, among others, by Adrien-Marie Legendre [27] and Carl Friedrich Gauss [19]. It has been a central topic of several books on functional equations, including [25] and [26] by Marek Kuczma. It is frequently referred to as the equation of homomorphism or the *Cauchy additive functional equation*, to distinguish it from other ‘Cauchy type’ functional equations:

- the Cauchy exponential functional equation

$$f(x + y) = f(x)f(y) \quad (2)$$

- the Cauchy logarithmic functional equation

$$f(xy) = f(x) + f(y) \quad (3)$$

- the Cauchy multiplicative functional equation

$$f(xy) = f(x)f(y) \quad (4)$$

(cf., [11], and [2] and [3] among others).

The *Pexider functional equations* are generalizations of the Cauchy equations mentioned above, when different unknown functions are allowed on each side of the equation ([34]):

$$f(x+y) = g(x) + h(y), \quad (5)$$

$$f(x+y) = g(x)h(y), \quad (6)$$

$$f(xy) = g(x) + h(y), \quad (7)$$

$$f(xy) = g(x)h(y). \quad (8)$$

The following functional equation

$$f\left(\frac{x+y}{2}\right) = \frac{f(x) + f(y)}{2} \quad (9)$$

is well-known as the *Jensen functional equation*. This equation characterizes functions that preserve the midpoint operation, which is strongly related to the concept of convexity.

The functional equation

$$f(x+y-xy) + f(xy) = f(x) + f(y) \quad (10)$$

is referred to as the *Hosszú functional equation*. It was introduced by Miklós Hosszú at the International Symposium on Functional Equations held in Zakopane, Poland, in 1967.

At the Symposium on Functional and Differential Equations held in Zawoja, Poland, in 1971, Jan Mikusiński considered the functional equation

$$f(x+y)[f(x+y) - f(y) - f(x)] = 0 \quad (11)$$

Later this equation was named after him.

If the function $f : \mathbb{R} \rightarrow \mathbb{R}$ is additive, then it also satisfies the following equation

$$[f(x+y)]^2 = [f(x) + f(y)]^2. \quad (12)$$

This immediately yields either $f(x+y) = f(x) + f(y)$, or $f(x+y) = -[f(x) + f(y)]$. Due to its form, it is referred to as an *alternative functional equation*.

The functional equation

$$f(x+y) + f(x-y) = 2f(x) + 2f(y) \quad (13)$$

is known as the *quadratic functional equation*.

The *d'Alembert functional equation* (cf., e.g., [17], [18], furthermore, [2] and [3]) named after the French mathematician Jean le Rond d'Alembert and inspired by the well-known trigonometric identity $\cos(x+y) + \cos(x-y) = 2\cos(x)\cos(y)$ is of the form

$$f(x+y) + f(x-y) = 2f(x)f(y). \quad (14)$$

The *inhomogeneous linear functional equation of order 1* has the form

$$f(ax+by+c) = Af(x) + Bf(y) + C, \quad (15)$$

where a, b, A, B are real numbers such that $abAB \neq 0$. (It was considered, e.g., in [1], [15], [16], [25] and [28].)

The trigonometric functions $f(x) = \cos x$ and $g(x) = \sin x$ satisfy the following functional equations

$$f(x+y) = f(x)f(y) - g(x)g(y), \quad (16)$$

$$f(x-y) = f(x)f(y) + g(x)g(y), \quad (17)$$

$$g(x+y) = g(x)f(y) + f(x)g(y), \quad (18)$$

$$g(x-y) = g(x)f(y) - g(y)f(x). \quad (19)$$

Hence equations (16)–(19) are called *trigonometric functional equations* (cf., e.g., [39]).

One of the most important special families of functional equations is the class of *(two variable) linear functional equations*

$$\sum_{i=0}^{n+1} f_i(p_i x + q_i y) = 0, \quad (20)$$

where n is a positive integer, $p_0, \dots, p_{n+1}$ and $q_0, \dots, q_{n+1}$ are real numbers, and $f_0, \dots, f_{n+1}$ are unknown functions. It has been considered in several papers and books (cf., e.g., [35], [37] and [38]). Several important functional equations belong to this class, among others, the Cauchy equation (1), the Pexider equation (5), the Jensen equation (9) and the quadratic equation (13) as a special case.

Another important class of functional equations is that of *Levi-Civita functional equations*

$$f(x+y) = \sum_{i=1}^n g_i(x)h_i(y), \quad (21)$$

where n is a positive integer, f, g_i, h_i , ($i = 1, \dots, n$) are unknown functions (cf., [36], [37]). This class also contains several well-known functional equations mentioned before. (For example, the Cauchy functional equations (1), (2); the Pexider functional equations (5), (6); trigonometric functional equations (16), (18).)

2. Description of the program

In this section, we describe the computer program, which determines the type of functional equations. More precisely, we will give detailed descriptions of three program functions `fewhattype`, `fetype`, `feinfo`.

Created in the computer algebra system Maple, the program is compatible with Maple 2021 and later versions. It was developed in a collaboration among the authors of the paper. A copy of its source can be obtained by contacting the

paper's first author. The program operates with formal computations, without numerical approximations, thereby precisely determining the types or classes of the given functional equations.

This property, using the concept of 'mathability' ([7]), 'increases the level of mathability' (or 'improves the mathability') of the underlying system (cf., also, [12] and [13]).

Three program functions, **fewhatype**, **fetype** and **feinfo** are contained in the Maple package **FunctionalEquations**. Similarly to other Maple programs, **fewhatype**, **fetype** and **feinfo** can be accessed with the forms

```
> FunctionalEquations[fewhatype](arguments);
> FunctionalEquations[fetype](arguments);
> FunctionalEquations[feinfo]();
```

or

```
> FunctionalEquations:-fewhatype(arguments);
> FunctionalEquations:-fetype(arguments);
> FunctionalEquations:-feinfo();
```

or, if the command

```
> with(FunctionalEquations);
```

has already been called in the current running Maple session, simply with

```
> fewhatype(arguments);
> fetype(arguments);
> feinfo();.
```

A detailed description of each function is given in the following part.

The function ***fewhatype***.

To run this function, at least one input parameter has to be given: the functional equations called **fe**. In addition, extra parameters **funcs** and **vars** can be used to highlight the names of the unknown functions and the variables of the functions contained. Thus, the input of the program should be of the form

```
> fewhatype(fe, [funcs, vars]);.
```

The list of the unknown functions **funcs** and the list of the functions' variables **vars** have to be given as a Maple list structure between square brackets, where all elements in the list are separated by a comma. By default,

```
funcs := [f, g, h, k, l];
vars := [x, y, z, u, v];.
```

Hence when users provide only the argument **fe**, the set of the unknown functions of the given functional equation has to be a subset of $\{f, g, h, k, l\}$. Similarly, the functions' variables must be in the set $\{x, y, z, u, v\}$. If not, users need to include additional parameters for the program to run properly.

Some examples of the correct form of input arguments are:

$$f(x+y) - f(x) - f(y) = 0, \text{funcs} = [f], \text{vars} = [x, y]$$

or simply

$$f(x+y) - f(x) - f(y) = 0$$

or

$$m(a+b) - n(a) - n(b) = 0, \text{funcs} = [m, n], \text{vars} = [a, b].$$

If the input is given correctly, the program will list all types and classes to which the given functional equation **fe** belongs. In the following, we present some examples with different options and their meanings.

Assume that the package **FunctionalEquations** has already been called in the current Maple session.

```
> fewhatype(f(x + y) - f(x) - f(y));
```

yields

Cauchy equation

Inhomogeneous linear functional equation of order 1

Linear type equation

Levi-Civita type equation

The program is able to simplify the given functional equation before starting the main process to determine its types or classes. For example:

```
> fe := sqrt(3).f(x + y) + sqrt(3).f(x - y) = sqrt(12).f(x) + sqrt(12).f(y);
```

```
> fewhatype(fe);
```

gives

Linear type equation

Quadratic equation.

When the unknown functions and their variables are not in the default case, users can add extra parameters to use the program, as follows:

```
> fe := s(a + b) = s(a).c(b) + c(a).s(b);
```

```
> fewhatype(fe, funcs = [s, c], vars = [a, b]);
```

yields

Levi-Civita type equation

Functional equation for trigonometric functions,
addition formula for sine.

The function *fetype*.

If users input a functional equation **fe** and specify a type or class of functional equations, this function will simply return the value **True** or **False** to indicate whether **fe** is the given type. Similarly to the previous program, extra parameters **funcs** and **vars** can be added as optional arguments to specify the list of unknown functions and their variables. Therefore, the input of the program should be

```
> fetype(fe , type , [funcs , vars]);.
```

Notes on the three parameters **fe**, **funcs**, and **vars** can be found in the previous subsection. The parameter **type** should be a string like given in the list of considered types and classes of functional equations mentioned in the second section.

For example,

```
> fetype(f(xy) - f(x) - f(y) = 0, Cauchy);
```

shows

```

      true;
> fetype(f(x + y) - g(x) - h(y) = 0, Cauchy);
```

gives

```
      false .
```

The program is able to simplify the considered functional equation before evaluating it, as follows:

```
> fetype(2.f((x + y)/2) = f(x) + f(y), Jensen);
```

yields

```
      true .
```

The extra parameter **funcs** is used in the example below to show the program's use when the unknown function is not contained in the default set:

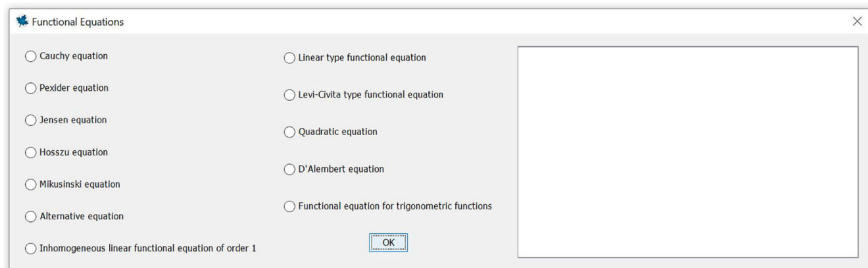
```
> fe := m(x + y - xy) + m(xy) = m(x) + m(y);
> fetype(fe , Hosszu , funcs = [m]);
```

gives

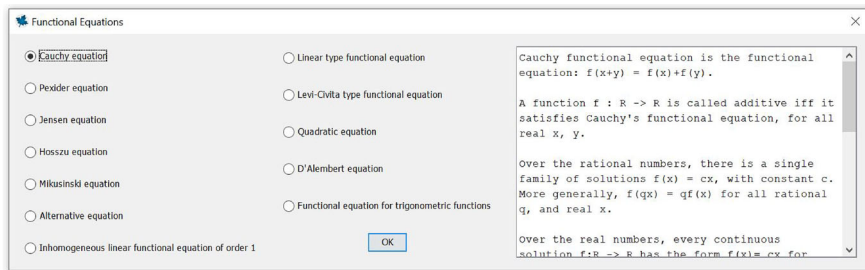
```
      true .
```

The function *feinfo*.

The function *feinfo()* is used to get information about all the considered types and classes of functional equations. When *feinfo()* is called, the following window will pop up



When one type or class of the functional equation is chosen, the corresponding information will be shown in the right-hand side textbox. For example, when the *Cauchy equation* is chosen, you will see



3. Explanation of the program

In this section, we show some procedure codes with detailed explanations to illustrate how our program works.

First, it can be observed that many types of functional equations share common terms. For example, the term $f(x + y)$ appears in the Cauchy functional equations (1), (2), the Pexider functional equations (5), (6), the Jensen functional equation (9), the quadratic functional equation (13), the class of Levi-Civita functional equations (21) etc. Similarly, the term $f(xy)$ is present in the Cauchy functional equations (3), (4), the Pexider functional equations (7), (8), and the Hosszú functional equation (10).

For this reason, our program checks the presence of common terms in the given functional equation and then determines its types or classes.

The following procedure checks the term $cf(x+y)$ where c is a constant in the given functional equation. Therefore, this procedure will be called later to check types and classes that contain the term $f(x+y)$, as mentioned above.

```

1  checkcfxplusy := proc(fe, var)
2      local Svar, flag, nfe, i, t, Lfe, Rfe, ufunc;
3      Svar := convert(var, '+');
4      flag := false;
5      nfe := fe;
6      for i to nops(nfe) do
7          t := op(i, nfe);
8          if type(t, function) and op(t) = Svar then
9              ufunc := op(0, t);
10             if not ufunc in FUNCS then
11                 error "Invalid unknown functions";
12             end if;
13             Lfe := t;
14             Rfe := Lfe - nfe;
15             flag := true;
16             break;
17         elif type(t, '*') then
18             if type(op(-1, t), function) and op(op(-1, t)) = Svar
19                 then
20                 ufunc := op(0, op(-1, t));
21                 if not ufunc in FUNCS then
22                     error "Invalid unknown functions";
23                 end if;
24                 Lfe := simplify(t/op(1 .. -2, t));
25                 Rfe := simplify(Lfe - nfe/op(1 .. -2, t));
26                 flag := true;
27                 break;
28             end if;
29         end if;
30     end do;
31     if flag = true then
32         return 1, Lfe, Rfe, ufunc;
33     else return 0;
34     end if;
35 end proc;
```

In this procedure, line 3 returns the sum expression of the unknown functions' variables, for example when $\text{var} = [x, y]$ then $\text{Svar} = x + y$. The program will run through all the terms in the given functional equation (line 6), then from line 7 to line 16, checks if the term $f(x+y)$ is there (it can be $g(u+v)$ or $m(x+y)$ depending on the unknown function ufunc and the list of variables var). Additionally, if the term appears as $cf(x+y)$, where c is a constant, lines 17–26 will not only recognize the term $cf(x+y)$, but the program will also cancel the constant c out of the given functional equation (lines 23–24). After checking all the terms contained, we have the procedure

- **return 0:** if the term does not appear;

- **return** 1, **Lfe**, **Rfe**, **ufunc**: term appears, where
Lfe: the appeared term,
Rfe: the rest terms of given functional equation
ufunc: unknown function in **Lfe**.

Similarly, the procedure `checkcfxtimesorxy` checks for the term $f(xy)$ and also the term $cf(xy)$, where c is a constant, in general. In mathematics the expression, xy and $x*y$ are both understood as the multiplication of two variables x and y . Therefore, two variables, **Mvar1** and **Mvar2** are used to ensure the program considers all the given cases. For example, if **var** = [**x**, **y**] then we have **Mvar1** = **x*y** and **Mvar2** = **xy**.

With the same structure and technique as in the previous procedure, the program will check each term of the given functional equation (line 41) in both cases, with and without the constant coefficient c 7 (lines 43–5) and lines 52–63, respectively). The procedure `checkcfxtimesorxy` returns the same output as the procedure `checkcfxplusy`.

```

35 checkcfxtimesorxy := proc(fe, var)
36     local Mvar1, Mvar2, flag, nfe, i, t, Lfe, Rfe, ufunc;
37     Mvar1 := convert(var, '*');
38     Mvar2 := cat(seq(var[i], i = 1 .. 2));
39     flag := false;
40     nfe := fe;
41     for i to nops(nfe) do
42         t := op(i, nfe);
43         if type(t, function) and (op(t) = Mvar1 or op(t) = Mvar2)
44             then
45             ufunc := op(0, t);
46             if not ufunc in FUNCS then
47                 error "Invalid unknown functions";
48             end if;
49             Lfe := t;
50             Rfe := Lfe - nfe;
51             flag := true;
52             break;
53         elif type(t, '*') then
54             if type(op(-1, t), function) and (op(op(-1, t)) =
55                 Mvar1 or op(op(-1, t)) = Mvar2) then
56                 ufunc := op(0, op(-1, t));
57                 if not ufunc in FUNCS then
58                     error "Invalid unknown functions";
59                 end if;
60                 Lfe := simplify(t/op(1 .. -2, t));
61                 Rfe := simplify(Lfe - nfe/op(1 .. -2, t));
62                 flag := true;
63                 break;
64             end if;
65         end if;
66     end do;
67     if flag = true then

```

```

66         return 1, Lfe, Rfe, ufunc;
67     else return 0;
68     end if;
69 end proc;

```

Let us consider all types of Cauchy equations (1)–(4) and Pexider equations (5)–(8). Their left-hand sides are considered by the two procedures above, and our program will check the rest of the terms, i.e., the right-hand side of the equation to give the conclusion. The procedure for this process is as follows

```

70 checkrhsCauchyPexider := proc(rhseq, var)
71     local t1, t2, ufunc;
72     if nops(rhseq) <> 2 then return 0; end if;
73     if not (type(rhseq, '+') or type(rhseq, '*')) then
74         return 0;
75     end if;
76     t1 := op(1, rhseq);
77     t2 := op(2, rhseq);
78     if not (type(t1, function) and type(t2, function)) then
79         return 0;
80     end if;
81     if evalb([op(t1), op(t2)] = var) or evalb([op(t2), op(t1)] =
82         var) then
83         if not (op(0, t1) in FUNCS and op(0, t2) in FUNCS) then
84             error "Invalid unknown functions";
85         end if;
86         ufunc := MakeUnique([op(0, t1), op(0, t2)]);
87         return 1, whattype(rhseq), ufunc;
88     else return 0;
89     end if;
90 end proc;.

```

Some conditions are checked for the right-hand side of the equation.

- Exactly two terms are contained (line 72) and the operator between them is addition + or multiplication * (lines 73–74);
- Both terms are in the form of a function (lines 78–80) and the unknown function's variables are correct (lines 81–87).

The procedure:

- **return 0**: when the given functional equation does not belong to any of the Cauchy equations (1)–(4) or Pexider equations (5)–(8);
- **return 1, operator, ufunc**: where
 - **operator**: the operator between the terms can be addition + or multiplication *;
 - **ufunc**: list of unknown functions, which is important for distinguishing between Cauchy types and Pexider types.

Lastly, with the procedure `feinfo`, we want to introduce a very surprising element of the Maple programming language – a graphic user interface (GUI).

An interactive window can be created with the function `Maplet` contained in the `Maplets:-Elements` Package.

```

90 feinfo := proc()
91     local maplet;
92     maplet := Maplet(Window('layout' = 'BL1', 'title' =
93         "Functional Equations"),
94         BoxLayout['BL1'](
95             BoxColumn('halign'='left',
96                 BoxRow(RadioButton['RB1']("Cauchy
97                     equation", 'group' = 'FE',
98                     Evaluate(#Cauchy equation's
99                     information))),
100             # other RadioButtons for different types
101             and classes of functional equations
102             Button("OK", Shutdown()))),
103         BoxColumn(BoxRow(TextBox['TB1']("", 'editable'
104             = 'false', 'width' = 50))),
105         ButtonGroup['FE']());
106     Maplets[Display](maplet);
107 end proc;

```

Z

In this procedure, simply a window pops up (line 99) which contains:

- Lines 95–96: a list of radio-buttons that allows users to choose their interested functional equation type or class,
- Line 98: a text box showing information on the chosen functional equation type or class,
- Line 97: a button OK simply for shutting down the application (i.e. closing the popped-up window).

Funding Open access funding provided by University of Debrecen.

Open Access. This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

- [1] Aczél, J.: Über eine Klasse von Funktionalgleichungen. *Comment. Math. Helv.* **21**, 247–252 (1948)
- [2] Aczél, J., *Vorlesungen über Funktionalgleichungen und ihre Anwendungen*. Birkhäuser, Basel; *Lehrbücher und Monographien aus dem Gebiete der exakten Wissenschaften*, p. 25. Mathematische Reihe, Bd (1961)
- [3] Aczél, J.: *Lectures on Functional Equations and Their Applications*. Mathematics in Science and Engineering, vol. 19. Academic Press, New York-London (1966)
- [4] Baják, Sz., Páles, Zs.: Computer aided solution of the invariance equation for two-variable Gini means. *Comput. Math. Appl.* **58**, 334–340 (2009)
- [5] Baják, Sz., Páles, Zs.: Computer aided solution of the invariance equation for two-variable Stolarsky means. *Appl. Math. Comput.* **216**(11), 3219–3227 (2010)
- [6] Sz. Baják and Zs. Páles. Solving invariance equations involving homogeneous means with the help of computer. *Appl. Math. Comput.*, 219(11):6297–6315, 2013
- [7] Baranyi, P., Gilányi, A.: Mathability: emulating and enhancing human mathematical capabilities. In *4th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 555–558. IEEE, (2013)
- [8] Borus, G. Gy., Gilányi, A.: On a computer program for solving systems of functional equations. In *4th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, page 939. IEEE, (2013)
- [9] Borus, G. Gy., Gilányi, A.: Solving systems of linear functional equations with computer. In *4th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 559–562. IEEE, (2013)
- [10] Borus, G. Gy., Gilányi, A.: Computer assisted solution of systems of two variable linear functional equations. *Aequationes Math.* **94**, 723–736 (2020)
- [11] Cauchy, A.L.: *Cours d’analyse de l’École Royale Polytechnique*. Première Partie. Analyse algébrique, Paris (1821)
- [12] Chmielewska, K., Gilányi, A.: Mathability and computer aided mathematical education. In *6th IEEE Conference on Cognitive Infocommunications (CogInfoCom)*, pages 473–477. IEEE, (2015)
- [13] Chmielewska, K., Gilányi, A.: Educational context of mathability. *Acta Polytechnica Hungarica* **15**, 223–237 (2018)
- [14] Czirbusz, S.: Testing regularity of functional equations with computer. *Aequationes Math.* **84**(3), 271–283 (2012)
- [15] Daróczy, Z.: Notwendige und hinreichende Bedingungen für die Existenz von nichtkonstanten Lösungen linearer Funktionalgleichungen. *Acta Sci. Math. (Szeged)* **22**, 31–41 (1961)
- [16] Daróczy, Z.: A bilineáris függvényegyenletek egy osztályáról (On a class of bilinear functional equations), (in Hungarian). *Mat. Lapok* **15**, 52–86 (1964)
- [17] d’Alembert, J.: Addition au mémoire sur la courbe que forme une corde tendue mise en vibration. *Hist. Acad. Berlin*, 6:355–360, (1750)
- [18] d’Alembert, J.: Memoire sur les principes de mecanique. *Hist. Acad. Sci*, pages 278–286, (1769)
- [19] Gauss, C.F.: *Theoria Motus Corporum Coelestium*. Hamburg, (1809)
- [20] Gilányi, A., To, L.N.: Computer assisted investigation of alienness of linear functional equations. *Aequationes Math.* **97**, 1185–1199 (2023)
- [21] Gilányi, A., To, L.N.: Computer assisted investigation of Levi-Civita type functional equations. *Manuscript*, (2025)
- [22] Gilányi, A.: Charakterisierung von monomialen Funktionen und Lösung von Funktionalgleichungen mit Computern. Universität Karlsruhe, Karlsruhe, Germany, Diss. (1995)
- [23] Gilányi, A.: Solving linear functional equations with computer. *Math. Pannon.* **9**(1), 57–70 (1998)

- [24] Házy, A.: Solving linear two variable functional equations with computer. *Aequationes Math.* **67**(1–2), 47–62 (2004)
- [25] Kuczma, M.: An Introduction to the Theory of Functional Equations and Inequalities, volume 489 of *Prace Naukowe Uniwersytetu Śląskiego w Katowicach*. Państwowe Wydawnictwo Naukowe–Uniwersytet Śląsk, Warszawa–Kraków–Katowice, (1985)
- [26] Kuczma, M.: An introduction to the theory of functional equations and inequalities. Cauchy’s equation and Jensen’s inequality. Birkhäuser Verlag, 2nd edition, (2009)
- [27] Legendre, A.M.: *Eléments de Géométrie*. Paris, (1791)
- [28] Losonczi, L.: Bestimmung aller nichtkonstanten Lösungen von linearen Funktionalgleichungen. *Acta Sci. Math. (Szeged)* **25**, 250–254 (1964)
- [29] Lundberg, A.: Some methods for a Sutô-Aczél project. *Aequationes Math.* **89**, 957–980 (2015)
- [30] Nadhomi, T., Okeke, C.P., Sablik, M., Szostok, T.: On a new class of functional equations satisfied by polynomial functions. *Aequationes Math.* **95**(6), 1095–1117 (2021)
- [31] Nadhomi, T., Okeke, C.P., Sablik, M., Szostok, T.: On a class of functional inequalities, a computer approach. *Periodica Mathematica Hungarica*, 1–13 (2025)
- [32] Okeke, C.P., Ogala, W.I., Nadhomi, T.: On symbolic computation of cp okeke functional equations using python programming language. *Aequationes Math.*, 1–20 (2023)
- [33] Okeke, C.P., Sablik, M.: Functional equation characterizing polynomial functions and an algorithm. *Results Math.* **77**(3), 125, 17 (2022)
- [34] Pexider, J.V.: Notiz über funktionaltheoreme. *Monatsh. Math. Phys.* **14**, 293–301 (1903)
- [35] Székelyhidi, L.: On a class of linear functional equations. *Publ. Math. Debrecen* **29**(1–2), 19–28 (1982)
- [36] Székelyhidi, L.: On the Levi-Civita Functional Equation. *Berichte der mathematisch-statistischen Sektion in der Forschungsgesellschaft Joanneum Graz, Graz* (1988)
- [37] Székelyhidi, L.: *Convolution Type Functional Equations on Topological Abelian Groups*. World Scientific Publishing Co., Inc., Teaneck, NJ (1991)
- [38] Wilson, W.H.: On a certain general class of functional equations. *Amer. J. Math.* **40**(3), 263–282 (1918)
- [39] Wilson, W.H.: On certain related functional equations. *Bull. Amer. Math. Soc.* **26**, 300–312 (1920)

Attila Gilányi
Faculty of Informatics
University of Debrecen
Egyetem tér 1
4032 Debrecen
Hungary
e-mail: gilanyi@inf.unideb.hu

Lan Nhi To
Doctoral School of Mathematical and Computational Sciences
University of Debrecen
Egyetem tér 1
4032 Debrecen
Hungary
e-mail: lan.nhi.to@science.unideb.hu

and

Institute of Mathematics and Computer Sciences
University of Nyíregyháza
P. O. Box 166
4400 Nyíregyháza
Hungary

and

Department of Mathematics and Computer Science
University of Dalat
Da Lat
Viet Nam

Received: April 5, 2025

Revised: June 10, 2025

Accepted: June 11, 2025