Proceedings of 4th International Conference CNDCGS'2024
ISSN 2669-2023 (print) / ISSN 2538-8959 (on-line) / DOI 10.3849/cndcgs.2024.288

Teaching in Software Applications: A Single Robust Built-In Command versus Sequence of More Elementary Commands

Vojtěch RŮŽIČKA[1*] Jiří JÁNSKÝ[1], Jan JEKL[1]

[1]Department of Mathematics and Physics, Faculty of Military Technology, University of Defence, Kounicova 65, 662 10 Brno, Czech Republic

Correspondence: [*]vojtech.ruzicka@unob.cz

Abstract

We investigate several exercises from the two fundamental disciplines of applied mathematics. We process the exercises in the software applications Maple and MATLAB: the exercises in Maple are from numerical analysis and the exercises in MATLAB are from statistics. For each example, we provide two ways of solution (not two different solutions) and try to explore their advantages and disadvantages for teaching students of university technical fields (undergraduate level). We pit solutions against each other: the first way of solution uses pre-prepared functions (robust built-in commands) for fundamental methods of given disciplines; in the second way of solution we have to "program" these methods by using sequences of fundamental commands. In the process, we theoretically establish parameter of education that needs to be carefully balanced for students' success. However, we are not only thinking about how well the students pass the subject if we use this or that way of solution. We also want to reflect on what we expect and what it is expected that students should learn. Which of the ways will provide them with a more "suitable" type of education for their further development? It seems that with advancing technologies (constantly expanding computing power, the current boom in AI) some of our positions about teaching in software applications need to be re-evaluated – is it really so, or is it just an illusion and the fundamental practices remain valid and they just adapt to new environments? Specifically, for us: Is it still necessary to understand what applications do, or is it enough to remember how to interpret their various outputs? Finally, let us note that we also pay attention to how challenging it is, even for us teachers, to find out what a given robust command actually calculates.

KEY WORDS: *built-in command; Maple; MATLAB; teaching applied mathematics; programming mathematical methods*

Citation: Růžička, V.; Jánský, J.; Jekl, J. (2024). Teaching in Software Applications: A Single Robust Built-In Command versus Sequence of More Elementary Commands. In Proceedings of the Challenges to National Defence in Contemporary Geopolitical Situation, Brno, Czech Republic, 11-13 September 2024. ISSN 2538-8959. DOI 10.3849/cndcgs.2024.288.

1. Introduction

We teach applied mathematics, especially numerical analysis and statistics. We dare to say that these mathematical disciplines are essential for a large number of university technical fields, the education process for national defence in not an exception, see [1] for an idea of what is included; however, we don't say that technicians are proficient in these disciplines. According to our experience, even long-solved fundamental problems are constantly being solved all over due to the lack of connection between abstract results and procedures used in practice. This may stem from the inappropriate approach of us, the teachers of these disciplines, to teaching.

Numerical analysis and statistics inherently need suitable software for teaching, but, and this is often forgotten, students need to have mastered the basics of the relevant "non-numerical" discipline of mathematics (the fact that students first pass some basic course does not often guarantee this).

Many articles have already been devoted to the teaching of mathematics on computers and their benefits are also often emphasized [2]. However, similarly to how calculators changed our approach to logarithmic tables, new technological means demand readjustment of current teaching methods, see [3]. However, it is natural that there is not one correct approach to teaching but a spectrum of possibilities, see [4,5,6]. And yet we could not find a scientific literature that investigates how necessary it is for students to understand the principle of built-in commands. And this very question will be studied in the following text: Is it necessary for students to have a clear understanding how any given command works?

Considering the focus of teaching at our school we only focus on how to perform some fundamental methods of numerical analysis and statistics in software applications (in the teaching process). Simply put, we are trying to explore the advantages and disadvantages of using pre-prepared functions (robust built-in commands) for fundamental methods of given disciplines over custom "programming" of these methods (just by using sequences of fundamental commands).

Peer-review under responsibility of General Jonas Žemaitis Miltitary Academy of Lithuania and University of Defence, Czech Republic

Depending on which way of solution we lean towards, it is more or less important, which application we use and also it is more or less important, if we use such application, which then professionals in that given technical field using. A related topic is already addressed in the older article [7], which notes the shift in the role of the computer in teaching mathematics: from an instructional tool (e.g. displaying graphic outputs; see also [8]) to a programming tool. Let us also note that we dealt with the categorization of the way software applications are used in the teaching process in our texts [4] and [5]. Further, let us emphasize that by pre-prepared functions we always mean functions built into an application, not finished user-made functions (scripts). Given this, we will also consider how important it is that we know exactly what the pre-prepared function calculates, i.e., we could program it by hand in any suitable software if needed, but we don't need to know precise implementation to the used application.

2. Methods of Investigation

We focus on pre-prepared functions in the applications Maple (for numerical analysis) and MATLAB (for statistics). We examine what a specific built-in function actually calculates; moreover, we will see how difficult it is, or it is not, to figure it out. Note that in [9] it is examined how Maple calculates the sum of an infinite series and how this can be useful for students. Finally, we will also provide an alternative to the pre-prepared built-in functions: the hand-written code to obtain the same result that the built-in functions provide. In fact, programming the method by hand is often the only practical way to find out what a built-in function actually calculates.

We can look at the matter from another point of view (more global), in which writing scripts goes hand in hand with the use of pre-prepared functions (both built-in and user-made). When solving a more comprehensive problem in practice (for students, for example, when writing a final thesis), it is inevitable that the student will look for solutions (user-made functions, a solution of the problem using built-in functions, etc.) on various web forums. He either find some existing solution or he directly asks. The ability to find and correctly adopt (modify) a solution is absolutely essential today, but be careful, we must not confuse this ability with being educated in the subject. If the student does not orient himself in the subject and only tries one found foreign solution after another, he will easily make a mistake and come to a completely meaningless conclusion, see, e.g., the section “Wrong interpretation of results” from [10]. Unfortunately, due to the described secondary nature, there is no time left for teaching how to find and apply foreign solutions (the student has to learn this ability himself). Materials (textbooks) with prepared code sequences, on the basis of which students create their code, can be a certain starting point for teaching, see [11]. We do not deal with the just-described problem of adopting user-made solutions from the Internet in this text, however, it is an important topic that is now even more relevant than before thanks to the rapidly growing capabilities of AI. Help from AI could even replace to a certain extent the simple search for solutions on the Internet – the question is whether AI is not an even more dangerous source of nonsense than the pure internet searching.

3. Commented Exercises from Numerical Analysis Solved in Maple

Exercise 1. Let us start with a simple exercise from Maple. For the approximate numerical calculation of definite integrals, so-called quadrature formulas are used. Maple has the robust command Quadrature in the package Student[NumericalAnalysis]. The command returns the approximate value of the integral corresponding to the selected formula type and other selected parameters if output is not set. Students should know from the lecture that, for example, in the special case of Newton–Cotes formulas, the interpolation polynomial is integrated, and they should therefore obtain the same result by manually entering the integration of the interpolation polynomial.

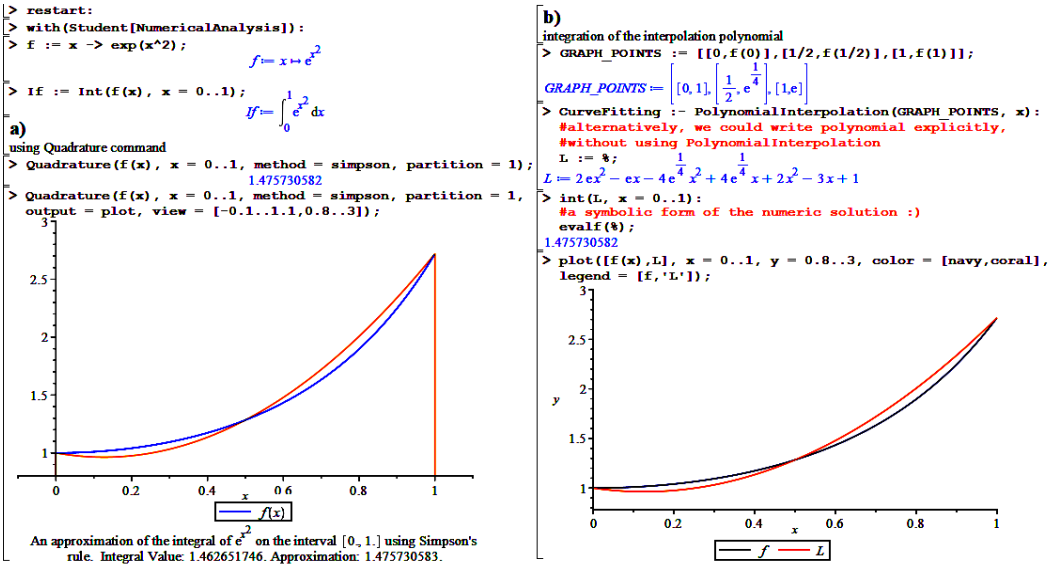


Fig. 1. Solution visualization: a) is more concise and easier for students to implement than the one in b).

The solution in Fig. 1 a) is more concise and easier for students to implement than the one in b). However, there may be a problem with setting the parameter view, more precisely, an explanation of why we set it in presented way (the command Quadrature itself would set the range on the y-axis from the minimum to the maximum value of the function f on the given interval). With the solution in Fig. 1 b), there will be no doubt about the meaning of the value 1.475730582 and what the orange line in the picture is, but even here it will be difficult to tell how the value 1.475730582 relates to the plotted graphs. However, in b), it will be "only" for ignorance of the basics of integral calculus; in a) for ignorance of the numerical integration. Note that if we wanted to numerically estimate the given integral as accurately as possible, we would use the command evalf(If), which uses a very good numerical method.

It should be added, that the code produced in Fig. 1 b) is specific to the quadrature formula used. As a consequence, the code presented in Fig. 1 b) would be diametrically different for other quadrature formulas. In general, we need to know how to determine the nodes and how to calculate the coefficients of the quadrature formula from them. So, it is up for consideration whether we require this level of knowledge from students. Additionally, the more code students write, the more it depends on how proficient they are with the application; common student mistakes in Maple and how to avoid them can be found in [12].

Exercise 2. We find the numerical solution of the equation $f(x) = 0$ on the interval $\langle a, b\rangle$. We assume that there is exactly one root on the specified interval. We use the false position method, to that we construct a sequence $\{x_k\}_{k\in\mathbb{N}\cup\{0\}}$, for which it holds

$$x_0 = a, \quad x_1 = b \quad \text{and} \quad x_{n+1} = \frac{x_m f(x_n) - x_n f(x_m)}{f(x_n) - f(x_m)} \quad \text{for} \quad n \in \mathbb{N},$$

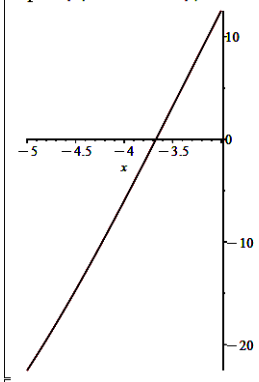
where $m = m(n) \in \{0,1,\dots,n-1\}$ is the biggest number such that $f(x_n)f(x_m) < 0$.

In Maple we use the command Roots with method = falseposition setting. The assignment of the function f and the numbers a and b is given in Fig. 2. We stop the iteration process after five steps, i.e., we have to compute the numbers x_2, x_3, x_4, x_5 and x_6 (five numbers); we do not choose the stopping criterion and the tolerance – it will become clear later why.

```
> restart:
> f := -x^3/2-5*x^2+2*x+50;
  a := -5.0:
  b := -3.0:
  k := 5: #k times the next iteration is calculated
```

$f := -\frac{1}{2}x^3 - 5x^2 + 2x + 50$

```
> plot(f, x = -5..-3);
```



a)
using Roots command

```
> with(Student[NumericalAnalysis]):
> Digits := 11:
  #to avoid differences caused by rounding
  Roots(f, x = [a,b], method = falseposition, maxiterations = k,
  output = information);
  Digits := 10:
```

n	a_n	b_n	p_n	$f(p_n)$	relative error
1	−5.0	−3.0	−3.7142857143	−0.787172010	0.19230769231
2	−3.7142857143	−3.0	−3.6719692815	−0.005491647	0.18299970125
3	−3.6719692815	−3.0	−3.6716741937	−0.000034058	0.18293403997
4	−3.6716741937	−3.0	−3.6716723636	-2.11×10^{-7}	0.18293363271
5	−3.6716723636	−3.0	−3.6716723523	$-2.\times10^{-9}$	0.18293363020

b)
creating procedure

```
> falseposition := proc(function, initial_approx, (iterations := 5))

  local k, f, a, b, p, R, n, i, RLIST, j, xm, tab:

  k := iterations:
  f := unapply(function,x):
  a[1] := initial_approx[1]: #x[0]
  b[1] := initial_approx[2]: #x[1]
  p[1] := (a[1]*f(b[1])-b[1]*f(a[1]))/(f(b[1])-f(a[1])): #x[2]
  R[0] := ['n','a[n]','b[n]','p[n]','f(p[n])','relative_error']:
   #the header of the table (zero row)
  R[1] := [1,a[1],b[1],p[1],f(p[1]),abs(p[1]-b[1])/abs(p[1])]:
   #first row

  for n from 2 to k do
     #finding x[m]:
     RLIST := [seq(p[i], i = 1..n-2),b[1],a[1]]:
     for j from 0 to nops(RLIST)-1 do
        i := j+1:
     until f(RLIST[i])*f(p[n-1]) < 0:
     xm := RLIST[i];
     #construction of the nth row:
     a[n] := min(p[n-1],xm);
     b[n] := max(p[n-1],xm);
      #if b[n] = p[n-1], then ERROR in Roots does not show up
     p[n] := (a[n]*f(b[n])-b[n]*f(a[n]))/(f(b[n])-f(a[n]));
      #x[n+1], n = 2,3,...,k
     R[n] := [n,a[n],b[n],p[n],f(p[n]),abs(p[n]-p[n-1])/abs(p[n])]:
      #ERROR in Roots: b[n] instead of the correct p[n-1]
  end do:

  tab := Matrix([seq(R[i],i=0..k)]):
  return tab:

  end proc:
> Digits := 11:
   #to avoid differences caused by rounding
  falseposition(f, [a,b], iterations = k);
  Digits := 10:
```

n	a_n	b_n	p_n	$f(p_n)$	relative_error
1	−5.0	−3.0	−3.7142857143	−0.787172010	0.19230769231
2	−3.7142857143	−3.0	−3.6719692815	−0.005491647	0.011524179413
3	−3.6719692815	−3.0	−3.6716741937	−0.000034058	0.000080368732200
4	−3.6716741937	−3.0	−3.6716723636	-2.11×10^{-7}	$4.9843771959\times10^{-7}$
5	−3.6716723636	−3.0	−3.6716723523	$-2.\times10^{-9}$	$3.0776166596\times10^{-9}$

Fig. 2. Comparison of the two ways of solution using one method of numerical root search

In Fig. 2 notice the blue tables below. Both tables for parts a) and b) have the first five columns exactly the same, so our code gives the same results as Maple's built-in procedure. Only the last column is different, but this is an error of the built-in procedure, not our procedure. We describe the implementation of the false position method into Maple to understand the meaning of the numbers in the tables and to describe why the values in the last columns are different.

Each table contains values a_1, b_1, p_1 ($n = 1$, first row); a_2, b_2, p_2 ($n = 2$, second row); a_3, b_3, p_3 ($n = 3$, third row) and so on. We match the values with our original sequence $\{x_k\}$. It holds

$$a_1 = x_0, \quad b_1 = x_1, \quad p_1 = x_2,$$
$$a_2 = \min\{x_{m(2)}, x_2\}, \quad b_2 = \max\{x_{m(2)}, x_2\}, \quad p_2 = x_3,$$
$$a_3 = \min\{x_{m(3)}, x_3\}, \quad b_3 = \max\{x_{m(3)}, x_3\}, \quad p_3 = x_4 \quad \text{and so on.}$$

Hence $p_n = x_{n+1}$ for $n \in \mathbb{N}$, where p_n is the approximate value of root obtained after n iterates (steps).

Now we describe (define) relative error from the tables. First note that relative error and tolerance together forms one type of stopping criterion for the false position method. Relative error is the number

$$\frac{|x_{n+1} - x_n|}{|x_{n+1}|}$$

for each $n \in \mathbb{N}$. If tolerance is ε, then the iterating process could be stopped by the stopping criterion

$$\frac{|x_{n+1} - x_n|}{|x_{n+1}|} < \varepsilon.$$

The error when calculating the relative errors in Fig. 2 a) is caused by the fact that the command Roots takes the value b_n instead of the value x_n. The value b_n is the maximum of the values x_n and $x_{m(n)}$ and therefore it is not always the required value x_n. Recall that $x_n = p_{n-1}$. Finally, let us note that there is an analogous problem when using absolute error.

Now let us focus on the didactic analysis of both ways of solution presented in Fig. 2. If we start to think about how the students will handle the presented ways of solution, the first impressions will probably be similar for everyone: method b) is difficult to impossible for students. We try to weaken this first impression so that we can even compare the two ways. Let us be aware that for the purposes of this article, method b) is as elaborate as possible and also as transparent as possible (for us), so that it describes the entire iteration process well, especially the calculation of relative error. Additionally, we created a table to match as closely as possible the table returned by the command Roots. It would be enough for students to enter the iteration relation in any way and after the calculation of each value also calculate corresponding relative error. Furthermore, let us note that the formula for the sequence $\{x_k\}$ seems too unapproachable without graphical interpretation of the false position method. Without a graphic idea of the method, all relations seem artificial and technical.

For the way of solution in Fig. 2 a) we very often encounter the fact that students have no idea at all what the meaning of the table is, and in fact they do not even understand the meaning of the individual rows and values in various columns of these rows are. And this despite the fact that we usually want graphic outputs after them (output = plot, output = animation) or even a simple output in the form of a sequence $\{[a_k, b_k]\}$ finalized by the value p_n (only when tolerance is reached). The graphical outputs (that might help especially visual learners in studying) are probably the biggest advantage of using the command Roots over writing own code. It would take us incomparably more work to render an image or even create an animation than just calculating a few elements of the iterative sequence $\{x_k\}$. The advantage of solution b) is that students will necessarily know how individual values are obtained. We dare to say that if students really independently process the calculation of elements of the iterative sequence, they will probably already know their meaning. When creating custom cycles, students must, at least in the end, find out where they need to get to. On the other hand, they do not need to know much about the meaning (graphical, for example) of the used numerical method. It is worth considering whether for students the ability to write their own code that calculate the method is almost as valuable as a thorough understanding of the method. Indeed, if student writes functional code, then he has already mastered most of the technical obstacles and understanding is already within reach.

Exercise 3. In the last demonstration in Maple, we will deal with the numerical solution of ordinary differential equations in the form

$$y' = f(x, y)$$

with the initial condition

$$y(x_0) = y_0.$$

We use the command InitialValueProblem from the package Student[NumericalAnalysis]. Let us note that Maple already has the command dsolve in the base, which can also construct numerical solution, but the command InitialValueProblem is much more suitable for teaching classical methods (Euler's method, Runge-Kutta methods and so on). However, if we wanted to find a high-quality numerical solution to the initial problem $y' = f(x, y)$, $y(x_0) = y_0$, we would use dsolve.

The syntax of the command InitialValueProblem is fairly easy to read with one major exception, which is the choice of method. We specify one particular method as an ordered pair [method, submethod]. If we do not specify submethod, the default one for the given setting of method will be used. Each method has its own default value of submethod. In the official help for the command, we learn which submethods are preset and what we should set if we want to use different method, e.g., for the modified Euler method we have to set method = rungekutta and submethod = meuler. But what we learn in the help to a minimal extent (rather not at all) is what is actually calculated for the selected pair – unfortunately, the name of the method does not always clearly and unambiguously specify the method (the formula). We can immediately ask ourselves whether we really need to know what is being calculated when using the built-in commands, and whether the students need to know this. Whatever the answer is, we think that at the very least teachers should have a good idea of what the command they are teaching actually calculates. Otherwise, it will easily happen that education in numerical methods will be reduced to just learning to alternate different pre-prepared solutions without any inner meaning.

To properly describe and analyze all method settings options (all pairs), i.e., to write which method is used together with the mathematical formula for calculation, would take at least the length of another article. When pairing the setup of pairs (in Maple) with numerical methods supplemented by mathematical formulas, which are e.g. well described in the textbook [13, section 7.2], we cannot avoid the fact that we do not program at least a few steps of the method ourselves. In

this text, we analyze only one one-step numerical method. Note that we would sweat a lot more with multi-step methods. If we were to analyze, for example, the method[adamsbashforthmoulton,step$k$], which is a multi-step predictor–corrector method, we would especially have to match the number k with the order of the predictor method and with the number of steps of the predictor method, and further with the order of the corrector method and with the number of steps of the corrector method. Some of the order numbers and number of steps match, but others differ – of course it depends on the definitions, e.g., the meaning of the method order usually differs between the explicit and implicit versions of the method. In addition, with multi-step methods, a one-step method must always be used as a starter (to perform the first steps, when we do not yet have enough values to perform a multi-step method). Fortunately, several mathematical formulas can be found in the official Maple help specifically for this predictor–corrector method, see [14].

In Fig. 3, we only set method = rungekutta. According to the help, it is preset submethod = midpoint. Suppose we do not have a textbook at hand where the midpoint method is described; moreover, even in textbook, the method can be named differently than in Maple. The midpoint method can be easily found on Wikipedia, see [15]. We immediately see mathematical formulas that we can use. However, we do not know if we want the explicit or implicit version: we assume that midpoint in Maple means the more usual explicit version. But beware, below we see "The explicit midpoint method is sometimes also known as the modified Euler method," (in Maple submethod = meuler) a further "Note that the modified Euler method can refer to Heun's method," (in Maple submethod = heun), see [15]. So, are these the same three methods, just specified differently, or are some two different? From the wording of the sentences on Wikipedia, we can conclude that the midpoint method (midpoint setting) and the Heun's method (heun setting) are two different methods, and the only ambiguity is which one is the modified Euler method (meuler setting). How wrong we would be! To make sure our reasoning is correct, let us try using the given formula and see if we get the same result as the one we get using the command InitialValueProblem with the method [rungekutta, midpoint], see Fig. 3.

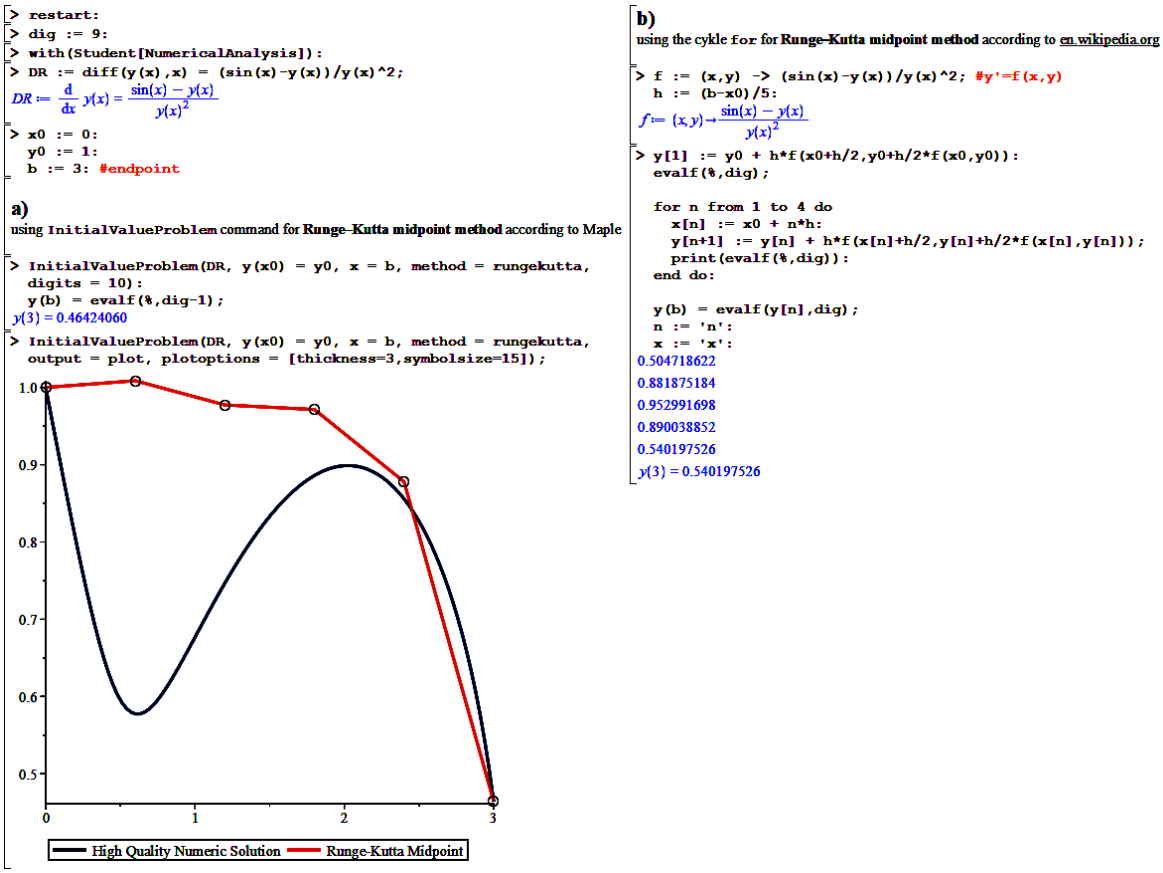


Fig. 3. Comparison of the two ways of solution using the midpoint method

In Fig 3, by comparison of the blue text with $y(3)$, we can see that the used methods in a) and b) are different. Even the graph in a) clearly says that $y(3)$ is lesser than 0.5 (ignore the different number of decimal places – the numbers from the built-in command are rounded differently for some reason). The graph is definitely one of the biggest advantages of the built-in command. We can create the graph ourselves, but it is inefficient if we mainly want to teach the numerical method.

Let us find out the correct formula for the settting submethod = midpoint. The command *InitialValueProblem* can compare methods. In Fig. 4 we can see that the midpoint method and Heun's method are the same, therefore try to use formula from Heun's method from Wikipedia [16], the result is in Fig. 4, on the right.

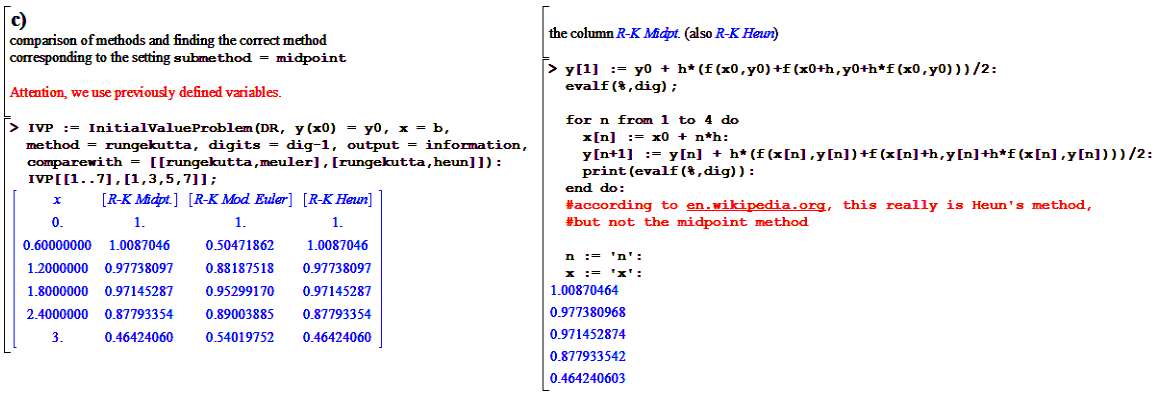


Fig. 4. Comparison of the midpoint method, Heun's method and the modified Euler's method

Finally, in Fig. 4, we can see that pairs (methods) [rungekutta, midpoint] and [rungekutta, heun] are the same with formula used in c) and the pair [rungekutta, meuler] is different method with the formula used in b) in Fig 3.

Now we compare the ways of solution. We could already see many advantages and disadvantages from the process of searching for mathematical formulas for methods from Maple. Let us just emphasize that if we were not able to write a simple cycle for to calculate the values, it would be difficult for us to find our way between the methods; the same applies to more complex multi-step methods. This is the clear difference between creating our own solution and strictly using the built-in command InitialValueProblem. For this reason, the second approach might be better for advanced students that are already familiar with Maple and the first approach might serve as an introduction for beginners. In this sense, as teachers, we only have to assess where we are already beyond the scope of what we want to teach, and vice versa, what else we would like to pass on to students. In fact, highlighting potential errors in the software might leave longer lasting impact on students, where they realize not to take everything as granted. Hence, this could lead to enforcement of students' critical thinking processes.

4. Commented Exercises from Statistics Solved in MATLAB

Exercise 4. We will now compare the use of the built-in function and the use of hand-written code using an example from mathematical statistics. Consider the *random samples* $X_1, X_2, \ldots, X_k$, where $k \geq 3$, $k \in \mathbb{N}$. Note that each random sample is a *random vector*, i.e., the vector of *random variables* with the same *distribution of probability* (briefly just *distribution*). The assignment of this exercise is to test the null hypothesis

$$H_0\colon \operatorname{med}_1 = \operatorname{med}_2 = \cdots = \operatorname{med}_k,$$

where med_i is the median (50% quantile) for the distribution of X_i for each $i \in \{1,2,\ldots,k\}$, against the alternative hypothesis H_1: H_0 does not hold. To test H_0 we use the Kruskal–Wallis test, which is well-known nonparametric test, see [17]. In order to perform this test, we must additionally assume that the random samples are *independent*.

Consider the random samples X_1, X_2, X_3, i.e. $k = 3$, and their realizations

$$\begin{aligned} x_1 &= (55, 54, 58, 61, 52, 60, 53, 65), \\ x_2 &= (52, 50, 51, 51, 49), \\ x_3 &= (47, 53, 49, 50, 46, 48, 50). \end{aligned} \tag{1}$$

Further denote $n_1 = 8$, $n_2 = 5$, $n_3 = 7$, which are dimensions (ranges) of x_1, x_2, x_3, respectively. Note that these values are usually retrieved from a file, but the values in Fig. 5 (a) are entered directly for transparency of this article. Note that the numerical values in (1) were taken from the exercise for Kruskal–Wallis test in [18, page 229].

First, we describe the use of the robust built-in command **`kruskalwallis`** from MATLAB. The input parameter of the command is the matrix in MATLAB of the type 8×3 (having 8 rows and 3 columns) containing the realizations x_1, x_2, x_3 as its columns, missing positions are filled with the symbol **`NaN`** (not a number). As teachers, we recommend that students use a comprehensive output in the form of a table, see Fig 5, variable **`tbl`**. The test can be easily evaluated by using the so-called p-value, which is listed in the last column in the row 'Groups'. It holds $p = 0.0012 < 0.05 = \alpha$, therefore at the significance level α we reject the null hypothesis H_0 in favor of the alternative hypothesis H_1 (this is the formal conclusion of the test). Thus, the testing by using the command **`kruskalwallis`** is quick and effortless. However, we as teachers have to think about whether this is enough for us, or whether we will want more from the students. We usually want to teach things at least in such depth that the students are forced to understand at least roughly "what it is about" and not just repeat teacher's solution. If we decide that we want to teach the internals of the test as well, we must prepare that the robust built-in commands will return different values than the ones we calculate using a sequence of elementary commands according to the formulas from the textbook. The ultimate solution would be to not use the robust command at all and write everything manually. In Fig 5 (b). we present the calculation of p-value, which returns the exactly same number as the command **`kruskalwallis`** from MATLAB. Let us strongly emphasize that getting the same value was difficult. The main problem here and also elsewhere is that the official documentation for the command **`kruskalwallis`**, see [19], usually does not contains explicit formulas for the calculation of the values of the output. Another problem was that the textbook we are using, see [18], does not contain the calculation correction, which in the end was the only one that really caused the difference between our manually obtained value and the value obtained from the command **`kruskalwallis`**.

```
clear
X1=[55 54 58 61 52 60 53 65];
X2=[52 50 51 51 49 NaN NaN NaN];
X3=[47 53 49 50 46 48 50 NaN];
X=[X1',X2',X3'];
[p,tbl,~] = kruskalwallis(X,[],'off');
tbl
```

tbl = 4×6 cell

	1	2	3	4	5	6
1	'Source'	'SS'	'df'	'MS'	'Chi-sq'	'Prob>Chi-sq'
2	'Groups'	466.5857	2	233.2929	13.4117	0.0012
3	'Error'	194.4143	17	11.4361	[]	[]
4	'Total'	661	19	[]	[]	[]

(a)

```
X = [X1',X2',X3'];
R = reshape(tiedrank(X(:)),size(X));
n = sum(~isnan(X));
N = sum(n);
k = size(R,2);
r = sum(R,'omitnan');
q = -3*(N+1)+12/(N*(N+1))*sum((r.^2)./n);
t = groupcounts(rmmissing(X(:)));
c = 1-sum(t.^3-t)/(N^3-N);
p = 1-chi2cdf(q/c,k-1)
```

```
p = 0.0012
```

(b)

Fig. 5. Research calculation description: (a) the values are entered directly for transparency; (b) the calculation of p-value, which returns the exactly same number as the command kruskalwallis from MATLAB.

As the code in Fig. 5 (b) is not transparent enough, we describe the calculation in more detail using mathematical formulas, see [18]. Assuming we already have the 8×3 matrix described above, let us denote it by $\boldsymbol{X}$ (the first, second and third columns are x_1, x_2 and x_3, respectively, not a number otherwise). All $N = n_1 + n_2 + n_3$ values must be sorted, regardless of which sample they belong to. We create the matrix $\boldsymbol{R}$ such that we replace numerical values in $\boldsymbol{X}$ by their order; the order will be described later. The matrix $\boldsymbol{R}$ has the successive columns

$$R_1 = (16, 15, 17, 19, 11.5, 18, 13.5, 20),$$
$$R_2 = (11.5, 7, 9.5, 9.5, 4.5),$$
$$R_3 = (2, 16.5, 4.5, 7, 1, 3, 7).$$

Let us describe the order of the repeating values. The value 50 is repeated three times and these are the seventh, eighth and ninth positions in the simple sorted vector of all numerical values of $\boldsymbol{X}$. However, the order of 50 for the test is 7; it is an arithmetic mean of theirs positions. The realization of the test statistic Q, see [18], is

$$q = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{r_i^2}{n_i} - 3(N+1) = 13.331, \tag{2}$$

where r_i is the sum of the values in the vector R_i for $i \in \{1,2,3\}$. The p-value is given by the relation $p = 1 - F(q)$, where F is the distribution function of the Pearson χ^2 distribution with $k-1$ degrees of freedom. By the command **`1-chi2cdf(q,k-1)`**, where **`k=3`** and **`q`** given by the formula (2), we obtain $p = 0.0013$. The result $p = 0.0013$ differs from that in Fig. 5. We must use the correction, which is described, for example, in [19]. Let $\{c_1, c_2, .., c_l\}$ be the set of all numerical values that appear more than once in the matrix $\boldsymbol{X}$. The correction term is

$$c = 1 - \frac{\sum_{i=1}^{l}(t_i^3 - t_i)}{N^3 - N} = 0.994,$$

where t_i is the number of repetitions of c_i for $i \in \{1,2,\ldots,l\}$. Clearly, instead of the set $\{c_1, c_2, .., c_l\}$ we can take the set of all values of $\boldsymbol{X}$, since $t^3 - t = 0$ if $t = 1$. For example, if $c_1 = 49$, then $t_1 = 2$, since the value 49 appears once in x_2 and once in x_3. Finally, the p-value is $p = 1 - F(q/c) = 0.0012237$. Note that if all the values in the input realizations are unique, then the correction term c is equal to one 1, and the values

$$p = 1 - F(q) \quad \text{and} \quad p = 1 - F(q/c)$$

are the same.

Finally, let's emphasize that we would never want students to reveal any "secret" formulas for built-in commands. They are not really secret, since MATLAB has accessible source code for each command, but it is indecipherable for an ordinary user (non-programmer). On other hand, if we know the formulas, it is not that hard to construct the matrix $\boldsymbol{X}$ (in MATLAB) or calculate realization q (in MATLAB), see Fig. 6. Each teacher should weigh the pros and cons of both ways of solution; and maybe select some ideal mix of the presented ways. We have to make sure that the student is not left untouched.

5. Conclusions

We solved several exercises in two ways (but using the same method), one way of solving usually relied on one robust built-in command for the used method, and in the other way the method was applied more explicitly by using several elementary commands. Based on these two ways, two corresponding directions of teaching can be distinguished. It could be

beneficial to examine how teachers of these and related mathematical disciplines approach these directions. The easiest implementation would probably be using a questionnaire (a survey). The respondents (teachers) would first familiarize themselves with the given exercises in order to have a good understanding of what is involved. Then they would be asked several questions in which it would find out how they teach and which of the partial options (from both ways of solution) seems more appropriate or correct to them. Overall, we suggest using the first approach for broader overview of methods and the second approach for deeper understanding.

Finally, let us point out that the requirements for military students can differ as opposed to civilian students. Furthermore, our results are theoretical in nature and empirical study might be necessary before practical implementation.

Acknowledgements

This research work was supported by the Project for the Development of the Organization 'DZRO Military autonomous and robotic systems'.

References

1. **Vagaská A., Račková P., Jekl J.** Key dimensions of an innovative approach to the teaching of mathematics at technical universities: Proceedings of the ICERI 2021, November 8–9, 2021, Online Conference.
2. **Xie C., Cheung A. C. K., Lau W. W. F., Slavin R. E.** The Effects of Computer-Assisted Instruction on Mathematics Achievement in Mainland China: A Meta-Analysis, *International Journal of Educational Research*, 2020, 105, 101565.

3. **Abou-Hayt I., Dahl B.** A Critical Look at the Laplace Transform Method in Engineering Education, *IEEE Transactions on Education*, 2023, doi: 10.1109/TE.2023.3285718.
4. **Jánský J., Jekl J., Růžička V.** Methods of teaching using software applications and true effects of these methods: Proceedings of the ICERI 2022, November 7–9, 2022, Seville, Spain.
5. **Růžička V., Jekl J., Jánský J.** Reflections on implementing software applications in education: utilizing a new categorization of teaching with applications, *accepted as a contribution to the book Changes and Innovations in Social Systems and Networks – A Bridge in Time*.
6. **Masegosa A. R., Cabañas R., Maldonado A. D., Morales M.** Learning Styles Impact Students' Perceptions on Active Learning Methodologies: A Case Study on the Use of Live Coding and Short Programming Exercises. *Education Sciences*, 2024, 14, 250.
7. **Aydin E.** The use of computers in mathematics education: a paradigm Shift from "computer assisted instruction" towards "student Programming". *The Turkish Online Journal of Educational Technology* 2005; 4, Article 4.
8. **Kilicman A., Hassan M. A., Said Husain S. K.** Teaching and learning using mathematics software "the new challenge". *Procedia - Soc. Behav. Sci.* 2010; 8, p. 613–619.
9. **Smith C. J., Tefera A., Zeleke A.** Computerized proof techniques for undergraduates. *International Journal of Mathematical Education in Science and Technology* 2012; 43, p. 1067–1077.
10. **Nečesal P., Pospíšil J.** Experience with teaching mathematics for engineers with the aid of wolfram alpha: Proceedings of the WCECS 2012, October 24–26, 2012, San Francisco, USA.
11. **Tucker M. C., Shaw S. T., Son J. Y., Stigler J. W.** Teaching statistics and data analysis with R. *Journal of Statistics and Data Science Education* 2023; 31, p. 18–32.
12. **Jánský J.** Teaching of mathematics in MAPLE program: Proceedings of the APLIMAT 2020, February 4–6, 2020, Bratislava, Slovakia.
13. **Stoer J., Bulirsch R.** Introduction to Numerical Analysis. 3rd edition. New York: Springer, 2010, xvi+752 p. ISBN 978-0-8218-1026-5.
14. AdamsBashforthMoulton. Available online: https://www.maplesoft.com/support/help/maple/view.aspx?path=Student/NumericalAnalysis/AdamsBashforthMoulton (accessed on 30. 4. 2024).
15. Midpoint method. Available online: https://en.wikipedia.org/wiki/Midpoint_method (accessed on 30. 4. 2024).
16. Heun's method. Available online: https://en.wikipedia.org/wiki/Heun%27s_method (accessed on 30. 4. 2024).
17. **Kruskal W. H., Wallis W. A**. Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association* 1952; 47, p. 583–621.
18. **Litschmannová M.** Introduction to statistics, Ostrava 2011. VŠB – TU Ostrava. (in Czech)
19. kruskalwallis. Available online: https://www.mathworks.com/help/stats/kruskalwallis.html (accessed on 30. 4. 2024).