Educational Technologies
Образователни технологии

COMPUTER PROGRAMMING IN MATHEMATICS EDUCATION

Marin Marinov, Lasko Laskov
New Bulgarian University (Bulgaria)

Abstract. In the last few decades, the contemporary informational technologies introduced new challenges in front of the education in mathematics. These challenges concern an essential part of the methodology of education itself. In this paper we present four different directions in which the important knowledge of computer programming may extend the capabilities of the educational process towards achieving the desired goals.

Keywords: mathematics education; computer programming; Wolfram language; Analytical Geometry; continuity; local extrema

Introduction

The incorporation of the contemporary informational technologies can turn from a threat into a tool for overcoming the crisis in the education in mathematics. This is the conclusion from the experience of New Bulgarian University in the education in some mathematical disciplines using a computer. The adoption of the contemporary informational technologies results in a new structure of education, which we call *practical oriented education*. The practical oriented education in mathematics achieves the educational goals using new forms, methods and means of teaching (Marinov, 2014).

The goal of the education is students to achieve:

– fundamental knowledge and skills, and ability to operate with them;

– skill to solve standard and nonstandard problems;

– skill to augment their knowledge by themselves.

A new educational type is developed in the form of computer labs, and the roles of lectures and practical classes is changed (Marinov, 2014). The three educational forms are implemented with a system of related problems in agreement with the presented approach for education based on a system of problems in (Asenova & Marinov 2019). This organization of the education places in the center of the process the learner himself, and the scientific software (in our experience this is the symbolic computation system Mathematica) plays the role of a tool for education and communication (Marinov, 2014).

We will note that the choice of the symbolic computation system Mathematica is due to its good didactic capabilities:

– for symbolic calculations in all mathematical disciplines;
– for visualization;
– for teaching proofs;
– for concepts introduction.

Besides that, the system is successfully used in scientific research and practical problems solution. Of course, there are other systems with the above qualities, and our choice of software system is more or less arbitrary, and is not a result of comparison between the existing systems. Also, in our work just a small part of the capabilities of the system Mathematica are used. We are convinced that the above methodology for teaching mathematics can be implemented without principal problems with the aid of the others software systems as well.

The adoption of the scientific software as a tool for education fundamentally changes the system of problems that are used in the educational process. A major part of the traditionally used problems can be solved directly in one step with one of the huge variety of build-in functions. More over, because of the variety of the build-in functions, many of the problems cannot play their previous part in the overall educational process. Such problems are: to draw the graph of a function, that is given in explicit, implicit or parametric form;
– to find the maximum (minimum) of a function on a compact domain;
– to calculate the derivative of a function;
– to calculate the integral of a function;
– to find the inverse or the pseudoinverse of a matrix, etc.

In this way the systems of problems, that has been approved by the good practices, are destroyed. Moreover, the insufficiency of problems for application of the studied material is significant. This insufficiency is overcame using the new capabilities of the scientific software. The good didactic capabilities of the system Mathematica allow to involve more natural, and at the same time more convincing problems, that help the introduction of the new knowledge. The path to the problems that are rich in content is getting much shorter (Marinov, 2008). The studying of proofs is getting possible (see (Marinov & Asenova, 2013)), which is a topic that is often neglected in university courses in mathematics in non-mathematical programs.

The integrated environment of the symbolic computation system Mathematica is comfortable for description of the relations between statements and concepts in different fields. In this way an opportunity is given for introduction of new problems that demonstrate the relations between the different studied subjects, and between different mathematical subjects as well. We will point out the relation to geometry, as an important one.

The adoption of the scientific software as a tool for education gives a good prerequisite to restore the relations of studying of mathematics with the external subjects. Traditionally, the relations to physics, economics, and biology are exam-

ined. In studying of mathematics by students in informatics programs, of course the relation to informatics is brought to the foreground. In this sense, the computer programming can be examined in two ways. The first one is programming as a field in which the knowledge of mathematics has its applications. With this paper we would like to draw attention to the other way – in which computer programming gives opportunity to include new types of problems to each of the systems of problems used in the education in mathematics. Something more, the well-structured such systems of problems give opportunity the skills and creativity that are developed by programming, to be included in education in mathematics. This makes the introduction of concepts much more effective (Asenova & Marinov, 2018), and the mathematical knowledge itself much more durable.

Computer programming in Mathematica

The symbolic computation system Mathematica introduces a comfortable integrated environment for programming in which easily can be applied the knowledge in different fields of mathematics. The programming language of Mathematica is called *Wolfram Language*. A computer program written in Wolfram Language is a sequence of commands, executed in interactive mode. Basic elements of the language include: numbers (integers, rational, real, complex), variables, main constants, arithmetical operators, comparison, logical operators, function identifiers, brackets. There is no symbol for end of expression, unless we want to execute the expression without printing – in this case semicolon is used.

Wolfram Language supports the following programming paradigm elements:

– symbolic computation (global and local transformation rules, usage of patterns in both global and local transformation rules);

– procedural programming (conditional statements, repeatedly executable statements, compound expressions, local variables);

– functional programming (definition of named, pure and anonymous functions).

The language itself has a comfortable syntax and allows complex programs to be written with relatively short program code (see (Wolfram, 2019)).

Computer programming in mathematical education

Even thought nearly all mathematical disciplines that are studied in the university can be pointed out, we will give four examples from the introductory courses in mathematics.

Graphics and animations

Graphics and animations are present in many of the problems that are used in the educational process. They allow the intuition in the understanding of abstract concepts and proofs. Their implementation in system Mathematica is made easier by the various build-in functions. What remains to be implemented is the connection of the appropriate build-

in functions with the mathematical knowledge into a short program. That is why there are two ways in which graphics and animations are effectively used. The first approach is when the lecturer gives the appropriate graphs and animations that illustrate the introduced content or during a numerical experiment that shows an idea, later developed in to a proof, etc. The second approach is when the students themselves implement the graphs by connecting the assimilated knowledge with their ability to write a short program. These type of problems are usually aimed in the development of their creativity.

Example 1. (*Continuity at a point.*) We define the function

$$f(x)=\begin{cases}\sin\left(\dfrac{\pi x}{2}\right)+\dfrac{1}{2} & \forall x<1\\ \dfrac{7}{4}, & x=1\\ -x^2+4x-1, & \forall x>1\end{cases} \tag{1}$$

(i) Using an animation in which the points x and x_0 can be moved on the plot of the function $y=f(x)$ (see Figure 1), we can attract the attention of the students to the specific features, that comprise the term continuity. For example, we select a point x_0, and we show the fact that in some cases $f(x)$ can be arbitrary close to $f(x_0)$, and in other cases - this is impossible. Something more, in the cases it is possible, the closeness is guaranteed by the closeness of x to x_0. In this way the conclusion is reached that for the examined function, in $x_0\neq 1$, the following property is true: $f(x)$ *can be arbitrary close to* $f(x_0)$, *as long as* x *is arbitrary close to* x_0.

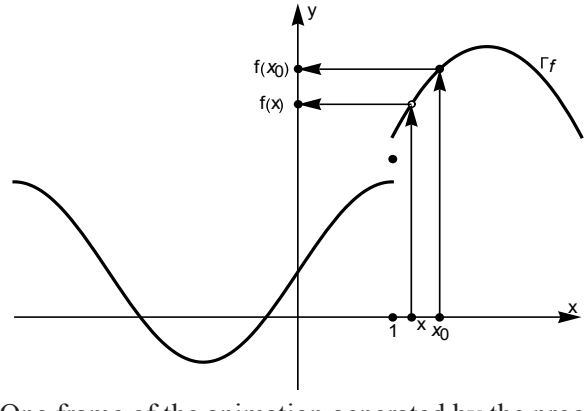

Figure 1. One frame of the animation generated by the program code 1

(ii) The animation helps to set aside the particular function. For this purpose, it is enough to analyze a number of different functions using the same program. Naturally we reach the definition of Koshi for continuity of a function in a point. A small complement to the initial animation allows us to illustrate it as well.

(iii) The full program code of the animation is given in Listing 1.

(1) Lines 1 and 2 define the function $f(x)$.

(2) Lines from 4 to 8 define the coordinates of the points of the animation.

(3) Lines from 9 to 25 generate the graph of the function and the locations of the labels in the animation.

(4) On the line 26 the animation itself is assembled.

```
H[a_, b_, x_] := Sign[x - a] + 1 / 2 * Sign[b - x] + 1 / 2;
f[x_] := (Sin[π * x / 2] + 1 / 2) * H[-4, 1, x] + (-x^2 + 4 * x - 1) * H[1, 5, x];
a = -3; b = 3;
X[x_] = {x, 0}; c[x_] = {x, f[x]};
d[x_, x0_] := If[x < x0, {0, Limit[f[t], t → x, Assumptions → t < x]},
  If[x > x0, {0, Limit[f[t], t → x, Assumptions → t > x]}, {0, f[x0]}]]
A[x_, x0_] := If[x < x0, {x, Limit[f[t], t → x, Assumptions → t < x]},
   If[x > x0, {x, Limit[f[t], t → x, Assumptions → t > x]}, {x0, f[x0]}]];
p[x_, x0_] := Plot[f[z], {z, a, b},
  PlotRange → {-0.8, 3.5}, AspectRatio → Automatic, Axes → False,
  PlotStyle → {Directive[Thick, Black]},
  Epilog → {
    PointSize[0.015], Point[X[x0]], Point[X[x]], Point[d[x0, x0]],
    Point[c[1]], Point[X[1]], Point[c[x0]], Point[d[x, x0]],
    {EdgeForm[Black], White, Disk[A[x, x0], Offset[2]]},
    Text["x", X[x] + {0.1, -0.08}],
    Text["x0", X[x0] - {0, 0.16}], Text["1", X[1] - {0, 0.16}],
    Text["f(x)", d[x, x0] + Sign[x] * {-0.27, 0}],
    Text["f(x0)", d[x0, x0] + Sign[x0] * {-0.27, 0}],
    Text["x", X[b] - {0.1, -0.1}], Text["y", {0.15, 3.4}],
    Text["Γf", {f[2.5] - 0.1, f[2.5] + 0.06}],
    Arrow[{A[x, x0], d[x, x0]}], Arrow[{X[x], A[x, x0]}],
    Arrow[{A[x0, x0], d[x0, x0]}], Arrow[{X[x0], A[x0, x0]}],
    Arrow[{{0, -1.5}, {0, 3.48}}], Arrow[{{a, 0}, {b, 0}}]
   }];
Animate[p[x, x0], {x, a, b}, {x0, b, a}, AnimationRunning → False]
```

Listing 1: Program that generates the animation to demonstrate the continuity at a point of a given function

Two similar animations together with \$10\$ graphics are used in (Asenova & Marinov, 2018) to present the separate stages of learning of mathematical concepts.

Teaching mathematical proofs

Let us clarify preliminary, that here we are not discussing *purely formal proofs*, that are subjects of the mathematical logic, and are represented by a sequence of symbols. The object of our research are so called *informal proofs*, which are taught in the courses of mathematics. These proofs create relations between the studied concepts. In this way, facts are transformed from being elements of a given set into a vital structure, that is capable to analyze and solve new problems. The knowledge and understanding of the main types of proofs, as well as the development of the ability for creating of individual proofs, are a basic stage in the education in mathematics.

In (Marinov & Asenova, 2013) is presented how all main types of proofs that are studied in the university can be taught using a computer. The examples that are shown in this work depict also the important role that takes the computer programming as an instrument for working environment organization.

We will illustrate the above with an example that uses the exhaustive search method.

Example 2. Find local extrema of the implicit function $z = f(x, y)$, that is defined by the equation:

$$x^4 - 2\left(x^2 + y^2 + z^2\right) + y^4 + z^4 = 0 \tag{2}$$

Solution: Since the function $u(x, y, z) = x^4 - 2\left(x^2 + y^2 + z^2\right) + y^4 + z^4$ has continuous second order partial derivatives, then the implicit function theorem gives the sufficient condition for the local existence of the implicit function $z = f(x, y)$. Simultaneously it is proved that the implicit function has continuous second order partial derivatives. As a corollary of the implicit function theorem the relation between the partial derivatives of the functions $u(x, y, z)$ and $f(x, y)$ is found. This allows to prove a procedure for finding local extrema of the implicit function $z = f(x, y)$, that is defined by the equation $u(x, y, z) = 0$.

(1) *Necessary condition*. Local extrema are stationary points. And in the examined case all stationary points can be found as solutions of the system

$$\begin{cases} \dfrac{\partial u(x, y, z)}{\partial x} = 0 \\ \dfrac{\partial u(x, y, z)}{\partial y} = 0 \\ u(x, y, z) = 0 \end{cases} \tag{3}$$

It will turn out that in the examined case that system has 19 solutions.

(2) We select those stationary points, for which the implicit function theorem can be applied. These are the points $A(x_0,y_0,z_0)$ for which

$$\frac{\partial u(x_0,y_0,z_0)}{\partial z} \neq 0 \tag{4}$$

It will turn out that in the examined case there are 18 such points.

(3) Verification of the *sufficient conditions* for local extreme in the stationary points for which (4) is fulfilled.

(3.1) In the examined stationary point $A(x_0,y_0,z_0)$, we calculate the matrix

$$m(x_0,y_0,z_0)=\begin{pmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{pmatrix}= \\ =\frac{1}{\dfrac{\partial u(x_0,y_0,z_0)}{\partial z}}\begin{pmatrix} \dfrac{\partial^2 u(x_0,y_0,z_0)}{\partial x^2} & \dfrac{\partial^2 u(x_0,y_0,z_0)}{\partial x\partial y} \\ \dfrac{\partial^2 u(x_0,y_0,z_0)}{\partial y\partial x} & \dfrac{\partial^2 u(x_0,y_0,z_0)}{\partial y^2} \end{pmatrix} \tag{5}$$

(3.2) We define functions

$$d_1(x_0,y_0,z_0)=f_{11} \qquad \text{and} \qquad d_2(x_0,y_0,z_0)=\det\begin{pmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{pmatrix} \tag{6}$$

(3.3) *Conclusion:*

(a) If $d_1(x_0,y_0,z_0)>0$ and $d_2(x_0,y_0,z_0)>0$, then the function $z=f(x,y)$ has a local minimum at the point (x_0,y_0), and it is equal to z_0.

(b) If $d_1(x_0,y_0,z_0)<0$ and $d_2(x_0,y_0,z_0)>0$, then the function $z=f(x,y)$ has a local maximum at the point (x_0,y_0), and it is equal to z_0.

(c) If $d_2(x_0,y_0,z_0)<0$, then the function $z=f(x,y)$ does not have a local extreme at the point (x_0,y_0). In this case we will call the point a *saddle point*.

(d) If $d_2(x_0,y_0,z_0)=0$ is necessary to perform a special investigation, because it is both possible function to have or not to have a local extreme.

```
v = {}; w = {}; n = {}; t0 = {}; t1 = {};
u[x_, y_, z_] = x^4 + y^4 + z^4 - 2 * (x^2 + y^2 + z^2);
uz[x_, y_, z_] = D[u[x, y, z], z];
m[x_, y_, z_] = -D[u[x, y, z], {{x, y}, 2}] // Expand / uz[x, y, z];
```

```
d1[x_, y_, z_] = m[x, y, z][[1, 1]] // Simplify;
d2[x_, y_, z_] = Det[m[x, y, z]] // Simplify;
t = Solve[
    D[u[x, y, z], x] == 0 && D[u[x, y, z], y] == 0 && u[x, y, z] == 0,
    {x, y, z}, Reals];
For[i = 1, i ≤ Length[t], i++,
 p = uz[x, y, z] /. t[[i]];
 If[p == 0,
  t0 = Join[t0, {t[[i]]}],
  q = {d1[x, y, z] /. t[[i]], d2[x, y, z] /. t[[i]]};
  If[q[[2]] > 0,
   If[q[[1]] > 0,
    v = Join[v, {t[[i]]}],
    If[q[[1]] < 0, w = Join[w, {t[[i]]}]]
   ],
   If[q[[2]] < 0,
    n = Join[n, {t[[i]]}], t1 = Join[t1, {t[[i]]}]]
  ]
 ]
]
```

Listing 2: Program that discovers local extrema of a given implicit function

The solution of the Example 2 with the above procedure requires a sufficient amount of routine calculations, and the application of computer programming skills make the solution easer and more complete. By using Mathematica's built-in functions for symbolic calculation of private derivatives, determinants, and system solving, the corresponding calculations can be performed by a computer. This allows the whole procedure for local extrema calculation to be implemented in a single program (see Listing 2).

In line 1 of the program in Listing 2 we define the lists in which the result will be stored, where:

– v is the list of points in which the implicit function has local minima;
– w is the list of points in which the implicit function has local maxima;
– n is the list of stationary points in which the implicit function has extrema;
– t_0 is the list of stationary points for which the implicit function theorem cannot be applied ((4) is not fulfilled);

– t_1 is the list of stationary points for which additional investigation is required.

In lines 2, 3, 4, 5, and 6 we introduce respectively: the examined function; its partial derivative with respect to the variable z; the matrix m (see (5)); functions d_1 and d_2 (see (6)).

In lines from 7 to 9 we calculate the list of the stationary points *t* by solving the system (3).

In the loop in lines from 10 to 24 we distribute the points from t into the lists v, w, n, t_0, and t_1. It turns out that for the concrete problem v and w contain five points each; *n* contains eight points; t_0 contains a single point, and t_1 is the empty set. More precisely:

$$v = \{(-1,-1,-\sqrt{\sqrt{3}+1});(-1,1,-\sqrt{\sqrt{3}+1});\ (0,0,\sqrt{2});$$
$$(1,-1,-\sqrt{\sqrt{3}+1});\ (1,1,-\sqrt{\sqrt{3}+1})\};$$

$$w = \{\left(-1,-1,\sqrt{\sqrt{3}+1}\right);\ \left(-1,1,\sqrt{\sqrt{3}+1}\right);\ \left(0,0,-\sqrt{2}\right);$$
$$\left(1,-1,\sqrt{\sqrt{3}+1}\right);\ \left(1,1,\sqrt{\sqrt{3}+1}\right)\};$$

$$n = \{\left(-1,0,-\sqrt{\sqrt{2}+1}\right);\ \left(-1,0,\sqrt{\sqrt{2}+1}\right);\ \left(0,-1,-\sqrt{\sqrt{2}+1}\right);$$
$$\left(0,-1,\sqrt{\sqrt{2}+1}\right);\ \left(0,1,-\sqrt{\sqrt{2}+1}\right);\ \left(0,1,\sqrt{\sqrt{2}+1}\right);$$
$$\left(1,0,-\sqrt{\sqrt{2}+1}\right);\ \left(1,0,\sqrt{\sqrt{2}+1}\right)\};$$

$$t_0 = \{(0,\,0,\,0)\}.$$

We will not that the presented solution of Example 2 turns a problem, that usually is omitted in the standard courses in Calculus, in a problem that helps to consolidate the acquired knowledge. Besides that, in this way the knowledge from another field (computer programming) is used to focus the attention on the analysis of the solution. More precisely:

– Geometrical interpretation of the solution: plot of the surface $S: x^4 - 2\left(x^2 + y^2 + z^2\right) + y^4 + z^4 = 0$ and the implicit functions $z = f(x,y)$; plot of the points from the lists v, w, n, and t_0 (see Figure (2)).

– Proof that the program solves the problem with an accent on the used theorems, where they are applied, and this application is possible.

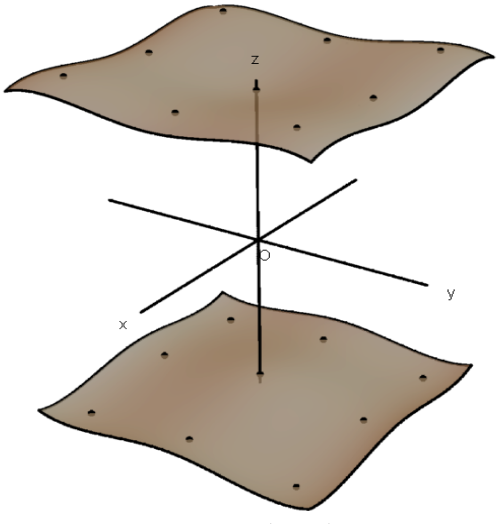

Figure 2. The implicit function $z = f(x, y)$ and the points of v, w and n

Our experience shows that the computer programming in the integrated environment of the system for symbolic computation Mathematica turns out to be a very effective tool: in numerical experiments with the goal of formation of hypothesis for a proof; in stating of some counter-examples; in illustration of decisive parts of the proof, etc.

Relation of mathematics with the real world and other sciences

Often solutions of applied problems presume a significant number of manipulations of the same type that lead to the final result. This places technical problems in the center of attention, and requires significant time for the solution. On the other hand, the giving up from the complete solution makes the relation to the real world declarative and unconvincing. The application of computer programming in such cases can be very useful. Examples for such problems can be many of the problems given in the standard Operations Research courses.

We will illustrate the above with a sub-problem of the 1D cutting-stock problem. The solution of the general problem leads us to a problem of the following type:

How many ways can be cut out of a pipe with a length of a *in pieces of lengths equal to any of the numbers* $q_1, q_2, ..., q_n$.

We will formulate the problem using only mathematical terms:

Example 3. *Division of a number by a vector.* Given a positive number a and a vector $q = (q_1, q_2, ..., q_n)$ with positive components, find all vectors $k = (k_1, k_2, ..., k_n)$, whose components are non-negative whole numbers, and satisfy:

$$0 \le a - (k_1 q_1 + k_2 q_2 + \cdots + k_n q_n) < \min\{q_1, q_2, \ldots, q_n\}$$

Solution: In the solution of Example 3, we will call the number a a *dividend*, and the vector q we will call a *divisor*. The vector k we will call a *quotient*, and the number $r = a - (k_1 q_1 + k_2 q_2 + \cdots + k_n q_n)$ – a *reminder*.

Without lost of generality, we will assume that a, and the components of the vector q are natural numbers, and also that $q_1 > q_2 > \cdots > q_n$.

For the problem formulated in this way, we can look for a solution that is motivated by the whole number division theorem. Because of the significant number of calculations of the same type, it is comfortable to solve the problem with an appropriate computer program. We will solve Example 3 by defining the function $F[a, q]$, which has arguments the dividend a, and the divisor q, and calculates the list v of all vectors k, which are quotients of the division of a by q.

By a sequence of numerical experiments students find a natural algorithm, that solves inductively the problem. For its implementation, first we define two helper functions. The first one is given in Listing 3 below:

```
fd[a_, x_] := Module[{A = a, b = x, v, i},
  For[v = {A}; i = 0, b ≤ A, i++, A = A - b; v = Join[v, {A}]];
  v]
```

Listing 3: Calculates the vector v

For given two numbers a and x, Listing 3 calculates the vector v with i -th component equal to $a - (i-1)x$ for $a - (i-1)x \geq 0$.

The second function is given in Listing 4:

```
fr[a_, x_] := Length[fd[a, x]] - 1
```

Listing 4: Calculates the whole part of the division of a by x

For given two numbers a and x, Listing 4 calculates the whole part of the division of a by x.

With these two functions we define the function $f[2, a, q]$ that solves Example 3 in the special case in which q has two components (Listing 5).

```
f[2, a_, q_] := Module[{A = a, b1 = q[[1]], b2 = q[[2]], v, w, i},
  w = fd[A, b1];
  For[v = {};
   i = 1, i ≤ Length[w], i++, v = Join[v, {{i - 1, fr[w[[i]], b2]}}]
  ];
  v]
```

Listing 5: Solves the example in the special case when q has two components

If we denote with $f[n, a, q]$ the solution in the case in which the vector q has $n > 2$ components, then the recurrent connection between $f[n, a, q]$ and $f[n-1, a, q]$ allows us to define the required function $F[a, q]$:

```
f[n_, a_, q_] := Module[{A = a, b1 = q[[1]], q1 = Drop[q, 1], v, w, m, m1, m2, i},
  w = fd[A, b1];
  For[v = {}; i = 0, i ≤ fr[A, b1], i++,
   m = f[n - 1, w[[i + 1]], q1];
   m1 = Array[i &, {Dimensions[m][[1]], 1}];
   m2 = Join[m1, m, 2];
   v = Join[v, m2]
  ];
  v];
F[a_, q_] := f[Length[q], a, q]
```

Listing 6: Recurent expression that defines the function $F[a, q]$

Defined in this way, the function $F[a, q]$ finds all vectors k, that are quotient of the division of a by q stores them as separate rows of the matrix v. Hence, the vector of the division remainders can be calculated using the function:

```
r[a_, q_] := a - F[a, q].q
```

Listing 7: Calculates the vector of the division remainders

In this case we have used the fact that if a is a number, and $u = (u_1, u_2, \ldots, u_n)$ is a vector, then Mathematica solves $a - u$ as a vector $(a - u_1, a - u_2, \ldots, a - u_n)$. For a given dividend, and divisor q, the function $r[a, q]$ calculates the vector $r = r[a, q]$ with components $r_j = a - v_{j\bullet}.q$. The number r_j is the remainder of the of the division of a by q with the quotient $v_{j\bullet}$, which is stored as the j -th element of the list v.

Finally, we will note that the number j of the vectors k, which are quotients of the division of the number a by the vector q, can turn out to be rather big. For example, for $a = 13$ and $q = (7, 6, 4, 3)$ it is $j = 10$. But for $a = 84$, and $q = (44, 32, 27, 15, 10, 8, 7)$, the number is $j = 303$.

Once Example 3 is solved, naturally the following problems arise:

By an appropriate modification of the solution of the example, define a function $G[a, q]$, that finds all vectors k, for which $0 \le a - (k_1 q_1 + k_2 q_2 + \cdots + k_n q_n)$.

Propose a more effective algorithm for the definition of the function $F[a, q]$.

Development of students' ability for mathematical knowledge application

The significance of this stage of education is getting more important because of the fact that in the recent practice the time scheduled for this type of problems was strongly limited. Problems that develop the ability of the students to apply mathematical knowledge can be such practical problems that require practitioner to discover the method for solving by himself. Another example are problems that require knowledge from different disciplines, or knowledge from different parts of the studied discipline, etc. Following this direction, computer programming allows the formulation of new types of problems. With these new problems, the attention of the students is attracted to the relations between the theorems that were already proved, and parts of proofs of different theorems. A new approach is developed towards the comprehension of the proofs in the general context of the education, and towards the development of the ability of the students for independent knowledge upgrade.

The following stages can be pointed out for solving of this new type of problems:

– Selection of the group of theorems, whose interrelations we will study.

– Definition of the interrelation that can be described in a formal way.

– Development of an algorithm for a computer program, that describes the interrelation.

– Proof of the result of the program execution.

– Program implementation.

– Program verification.

– Application of the implemented program.

As examples can be pointed out many analytical geometry statements and theorems that concern: equation of a line in the Euclidean plane and the Euclidean space; determining the mutual position of two straight lines in space; classification of second degree lines and surfaces, etc. We will illustrate the above with a short example.

Example 4. Write a program that determines the mutual position of two arbitrary planes in space, if the planes are defined with their general equations.

Solution: For two planes in space, exactly one of the cases is true:

1. The two planes intersect.
2. The two planes are parallel.
3. The two planes coincide.

By condition, the two planes are defined by their general equations:

$ax + by + cz + d = 0$ and $ex + fy + gz + h = 0,$

where a, b, c, d, e, f, g, h are predefined constants, for which $|a| + |b| + |c| \neq 0$ and $|e| + |f| + |g| \neq 0$. We will define a function $F[\alpha, \beta]$, that will determine which of the three possible cases is valid for planes with general equations $\alpha = 0$ and $\beta = 0$.

Following the general approach, first we determine the group of theorems that give the full description of all possible cases of mutual position of two planes in the space. After that, we determine those concepts which allow the uniform representation of the

conditions of the theorems, without violation of the truthfulness of their proofs. This allows us to discover a logical relation that leads to the development of the algorithm, and hence, to the solution of the problem. For the specific problem, the algorithm is:

1. Using α, define the vector $v=(a, b, c, d)$.
2. Using β, define the vector $w=(e, f, g, h)$.
3. Calculate the rang of the matrix $m=\begin{pmatrix} a & b & c & d \\ e & f & g & h \end{pmatrix}$.
4. Calculate the rang of the matrix $m=\begin{pmatrix} a & b & c \\ e & f & g \end{pmatrix}$.
5. If the rang of the matrix m_1 is equal to 2, then the planes intersect.
6. If the rang of the matrix m_1 is different from 2, and the rang of the matrix m is equal to 2, then the planes are parallel.
7. If both rangs of m_1 and m are different from 2, then the planes coincide.

The proof of the algorithm actually repeats the proof of the corresponding theorems. A possible implementation of the algorithm in Wolfram Language is given in Listing 8.

```
F[α_, β_] :=
 Module[{v, w, m, m1, r, r1, p},
  v = {Coefficient[α, x], Coefficient[α, y], Coefficient[α, z], α /. {x → 0, y → 0, z → 0}};
  w = {Coefficient[β, x], Coefficient[β, y], Coefficient[β, z], β /. {x → 0, y → 0, z → 0}};
  m = {v, w};
  r = MatrixRank[m];
  m1 = Take[m, All, 3];
  r1 = MatrixRank[m1];
  p = If[
    r1 == 2, "The planes intersect",
    If[r == 2,
     "The planes are parallel", "The planes coincide"
    ]
   ];
  p]
```

Listing 8: Determines the mutual position of two arbitrary planes in space

Remark: The program in Listing 8 can be modified not to use the build-in function *MatrixRank*[...], but to use only algebraic expressions of the constants a, b, c, d, e, f, g and h. In this way it is not going to be necessary to define the matrices m and m_1, and to apply the function *Take*[...].

Conclusion

The purposeful abruption of the relation of mathematics with the real world, that started in the middle of 20th century, substitutes the essence of mathematics with a training of

formal manipulations (Arnold, 1990). This leads to a wrong concept of mathematics, and eventually leads to its destruction. The contemporary informational technologies, applied in the solution of mathematical problems, cause new treads, that result in acceleration of these negative trends.

On the other hand, the scientific software, when adopted as a tool for education, can help to overcome the wrong concept of mathematics, and to avoid the new threats, by putting back into the center of education the meaningful aspect of mathematics. From this point of view, the initial programming skills turn out to be very useful in:

(i) application of geometrical interpretations in education;
(ii) understanding of meaningful mathematical proofs;
(iii) education in mathematics that is focused on the practice.

The inclusion of computer programming in the methods of education in mathematics:

(i) enhances the students' ability for application of mathematical knowledge;
(ii) increases the interest of students for the particular problem, and increases their activity;
(iii) leads to the development of new category of problems;
(iv) develops students' algorithmic thinking;
(v) consolidate students' programming skills.

REFERENCES

Arnold, V. (1990). The antiscientifical revolution and mathematics*, Herald of the Russian Academy of Sciences* 69: 553 – 558.

Asenova, P. & Marinov, M. (2018). Teaching mathematics with computer system, *Mathematics and education in mathematics*. *UBM 47*: 213 – 220.

Asenova, P. & Marinov, M. (2019). System of tasks in mathematics education, *Mathematics and Informatics* 62: 53 – 71.

Marinov, M. (2008). *Matrix calculation with Mathematica*, Sofia: Planeta 3.

Marinov M. & Asenova., P. (2013). Mathematical Proofs at University Level. *Computer Science and Education in Computer Science, Fulda, Germany*, 72 – 81, ISSN1313624.

Marinov, M. (2014). Mathematical education with system for symbolic calculation, *Mathematics and education in mathematics*. *UBM* 44: 137 – 148.

✉ **Prof. Dr. Marin Marinov**
✉ **Dr. Lasko Laskov, Assoc. Prof.**
Department of Computer Science
New Bulgarian University
21, Montevideo Blvd.
1618 Sofia, Bulgaria
E-mail: mlmarinov@nbu.bg
llaskov@nbu.bg