Overview of some technologies for designing a task generator in higher mathematics for distance learning systems

S.A. Mukhanov[1,2*], *A.A.* Mukhanova[3], and *V.V.* Britvina[1]

[1]Moscow Polytechnic University, 107023, Bolshaya Semyonovskaya str., 38, Moscow, Russia
[2]MIREA - Russian Technological University, 119454, Vernadsky Avenue, 78, Moscow, Russia
[3]Moscow Region State University, 141014, Mytishi, Very Voloshinoy, 24, Moscow Region, Russia

Abstract. Successful teaching of mathematics to students is impossible without their independent performance of practical tasks, the compilation of which manually is very laborious. This article provides an overview of some technologies for designing task generators in higher mathematics, from the point of view of both those algorithms that can be used to generate tasks, and from the point of view of the technologies used to build the generator. There are also some options for implementing task generators: based on the Microsoft Office office suite, a Python program, an online JavaScript generator. In all the proposed variants of the generator, computer layout systems (or a translator) are responsible for translating mathematical formulas into a readable form TeX (or LaTeX). The generator based on the Microsoft Office suite uses such features of the office suite as random number generation, branching using the "IF" function of Excel and using the merge tool in Word. The proposed generator, written in Python, uses recursive functions to generate tasks on the topic "Derivative of a function" of various types. The generation of the type of tasks on the topic "Integrals" is carried out taking into account the selected canonical forms. The proposed online JavaScript generator uses similar concepts. The specified generator can be effectively integrated into the LMS MOODLE, which allows it to be used for building distance courses.

1. Introduction

An important component of teaching students mathematics is the independent execution of variants of tasks of a typical calculation. To do this, the teacher compiles works divided into certain sections of the curriculum. In this case, the student may discover any features of the solution that are not visible in ordinary examples and could have been overlooked by the teacher at lectures. In addition, repetition during training and development of the necessary skills, according to the Ebbinghaus forgetting curve, allows you to achieve the necessary result in practice qualitatively and confidently [1].

The development of a large number of tasks requires a significant amount of time from the teacher, so the problem of generating tasks is regularly raised in pedagogical science [2-

* Corresponding author: s_a_mukhanov@mail.ru

5]. When developing a task generator for a higher mathematics course, it is necessary to take into account the complexity of a set of mathematical formulas. Works of Gangur, M. or Gladavská, L. and Plevný, M., for example, are devoted to these issues [6,7]

To solve this problem, you can use the TeX or LaTeX computer layout system to prepare formulas, and entirely for the text of standard calculations. Various task generators are also built on the capabilities of this system. [8,9]. This system allows you to prepare the content of a typical calculation, including complex mathematical formulas, in plain text. This concept allows you to effectively use the functions for working with strings in programming languages to generate the corresponding text files, which then, when processed in computer layout systems, allow you to get a PDF file with correctly displayed mathematical formulas.

The works of M.Gangur et al. are devoted to the development of various task generators for testing systems that also use TeX syntax for marking formulas, but are initially focused on web technologies using MoodleTeX in LMS MOODLE or jsmath, mimeTeX in other testing systems [10-12].

In addition to being used in computer layout systems, the generated formulas can be found on pages on the Internet, for example, when using the KaTeX library. This library allows you to use formula generation using, for example, JavaScript. Its advantage is that it has no dependencies on other libraries and provides high speed in all modern browsers. In addition to this method, there are also many other ways to integrate LaTeX into HTML. For example, using a simple LaTeXMathML library.js in JavaScript or an open source MathJax cross-browser library released under the Apache license, which allows you to display mathematical formulas in browsers using MathML, LaTeX and ASCIIMathML markup.

The purpose of this article is to consider different mechanisms for solving problems of generating tasks in mathematics, which can be used, including for distance learning systems such as MOODLE. The proposed solution to this problem can be an online exercise generator using modern web application development standards.

2 Methods

The most important and responsible stage of development is the design of the future application. At this stage, the interface is formed, the future user experience is formed, the main stages of the application's business processes are investigated.

2.1. General approaches to generation algorithms

We will use partial-template generation. At its core, each mathematical expression is a set of component parts in which the numerical coefficients and the configuration of the parts ("template") change relative to the indirect calculated coefficients ("meta-coefficients").

In view of the chosen topic, we will add and form such templates based on a random selection from the list, from which, if necessary, tabular values are excluded, for example, which do not lead to the formation of the required skills in students.

Further, the selected templates in connection with random numerical coefficients should be converted into a set of TeX commands, which will be further converted to PDF, HTML, SVG or MathML for display in any modern browser.

2.2. Generation of a permanent individual set of tasks for each student

The task generator can be powered by a random number generator. In this case, every time the generator is started, we will receive new tasks, which, in many aspects, is extremely inconvenient, because it does not allow the teacher and the student to receive the same sets

of tasks. Thus, the teacher must generate a variant and then transfer a copy to the student, which is inconvenient or impossible when placing the generator itself on the site.

Taking into account the above, it seems to us that an important point in the design of the generator is the possibility of obtaining absolutely identical sets of tasks when entering the same data set. Thus, it is necessary to provide for the possibility of generating a certain set of data (usually numeric) when the user enters, for example, his full name and number or group name, i.e. generating some hash function that will, in fact, determine the task option, all sets of functions and coefficients used for generating.

The issue of generating such a function, depending on the input parameters that we propose to make the student's personal data, does not seem difficult and we will not consider it.

2.3. Possible differences in the algorithms for generating tasks in different sections of higher mathematics

In relation to different branches of mathematics, the methods of generating tasks may differ significantly from each other. For example, to generate many tasks in linear algebra and analytical geometry, students are most often asked to solve problems related to operations on vectors or matrices and their properties. Therefore, it is quite simple to generate coefficients in equations, coordinates of points, etc. Here, often, there is not even a special need to control the values obtained. You may have to control only the very possibility of solving the problem with the specified coefficients. For example, the very existence of a figure or body defined by the coordinates of its vertices. And, as it seems to us, even this is not necessary in many educational tasks, because a student should be able to work in non-standard situations, solve non-standard tasks or see the impossibility of solving them and should be able to justify the impossibility of solving (or unambiguous solution) a formulated problem with errors in the condition.

In some sections of mathematics, this approach is absolutely inapplicable, for example, when designing a generator on a topic related to finding integrals, it must be borne in mind that we can not integrate every elementary function. For each type of indefinite integral, it is necessary to create a highly specialized template that allows you to work out each solution method and at the same time take into account the very possibility of using the integration method being worked out. For example, if in the task it is necessary to take an integral using the variable replacement method, then it must be taken into account that in order to implement this method, in the case of replacement $t=f(x)$, the function $f'(x)$ must also be included in the task [9].

In the same series there will also be tasks on the topic "Differential equations", primarily because the solution of such tasks is often again reduced to finding integrals. The only possibility here is to highlight interrelated canonical forms of tasks (for example, the relations $f(x)$ and $f'(x)$ in the example discussed above), which guarantee the possibility of solving the generated examples.

Among the many branches of mathematics, in terms of generating tasks, the section related to finding derivatives stands out somewhat. Its peculiarity is due to the fact that the derivative can be taken from almost any function, i.e. there is no need to use what we called canonical forms of writing tasks just above. At the same time, unlike tasks in the sections of linear algebra and analytical geometry, not just coefficients should be generated here, namely complex functions of different levels of nesting.

2.4. Generating answers to tasks or generating test tasks

This article discusses a task generator for practicing problem solving skills, i.e. for students to solve them independently, and not for the implementation of the control function. If you need to get solutions in addition to tasks, or if you need to prepare a set of tasks to monitor the students' assimilation of the material learnt, then you will need to supplement the generator with some modules or use third-party solutions. A huge number of solutions on the Internet, such as Wolfram Alpha, allow you to solve complex tasks accurately and have a flexible API, but it does not have the ability to generate tasks or this capability is very poorly expressed [13]. Therefore, our generator will create a set of tasks on the selected topic, and the answers to the generated tasks will be calculated either using a third-party API or libraries.

If necessary, since it is the author who determines the algorithm for generating tasks, that is, the author initially knows the algorithm for solving it, which leads to the fact that it is easy to supplement the generator with a module that allows you to get an answer to the task, which can help in the formation of a closed methodological section for students containing answers and / or instructions to the solution.

Currently, systems for assessing students' knowledge using a test are being intensively developed. In test tasks, in addition to the correct answers, it is often necessary to generate a number of incorrect answers. To implement this task, in our opinion, it is necessary to analyze the typical mistakes that students make when solving this type of problems. Thus, our proposed approach partially echoes the ideas considered by Goguadze and Zinn [14,15]. The result of this work can help in the development of the generator module responsible for compiling incorrect answers.

Let's explain what we mean when we talk about the possibility of generating correct and incorrect answers. For example, if we need to design a generator on the topic "Derivative of a complex function", then our generator forms a complex function by putting one or more functions into one another, like nesting dolls (we described the process in more detail in this article below). Knowing which function is used at each step, we certainly know both its derivative and its arguments, so we can easily get the correct answer to this task simply by correctly applying the formula of the derivative of a complex function when generating the answer (without simplifications). To generate incorrect answers, we note that when calculating the derivative of a complex function, students often make the following error: the derivative of each nested function is calculated correctly separately, but the result is written as a complex function composed of the derivatives obtained, i.e. the derivative of the previous one acts as the argument of the derivative of a later action, which, of course, is absolutely incorrect. Given this fact, we can easily make up an appropriate incorrect answer. This is one of the possible variants of a typical error. Analyzing the tasks and considering other typical mistakes of students, we can compile a set of incorrect answers to generate closed-type test tasks.

3 Findings

The practical implementation of the methods discussed above for the design of task generators for the course of higher mathematics can be implemented by various technical methods.

3.1. The general principle of building generators

All the generators we have discussed below work according to the same general scheme:

- generation of a set of random (or pseudo-random when using the hash function, which we talked about above) values that determine both the functions themselves used when writing mathematical examples, and the coefficients in these functions.
- generating tasks depending on the specific values obtained in paragraph 1.

Let's consider the specific results of applying the generation methods discussed above.

3.2. Task generator on the topic "Indefinite integral" based on Microsoft Office suite applications

Our development was based on the possibility of generating random numbers in Excel, depending on the value of which a function was selected in LaTeX notation.

Since, as mentioned earlier, the dependence between functions that should eventually form an integral expression is important for integrals, we needed to use chains of dependent cells with attachments, which was implemented using the "IF" function. In particular, when generating the integral of the function sin(g(x)), it was necessary to make this function multiply by the derivative of g(x), i.e. the expression must necessarily be obtained: sin(g(x))g'(x). Of course, in order to hide the explicit form of the derivative of the function g(x) in the process of its generation, other coefficients were used, as well as some other methods.

An example of one of these functions is given below:

=IF(Sheet 1!J306=1;CONCATENATE("\sin^";TEXT(H307;"#");"{";
TEXT(R307;"#");"x}\cdot"); IF(Sheet 1!J306=2;
CONCATENATE("\frac{\tg^"; TEXT(E307;"#"); "{";TEXT(R307;
"#");"x}}"); IF(Sheet 1!J306=3; CONCATENATE("\frac{\arctg^";
TEXT(F307;"#"); "{"; TEXT(R307;"#");"x}}"); IF(Sheet 1!J306=4;
CONCATENATE("\frac{\arcsin^"; TEXT(J307;"#"); "{";
TEXT(R307;"#");"x}}")))))

As you can see, formulas in TeX format are already used here.

The data obtained using the merge method was inserted into the Word text editor, where the text of the variant using LaTeX was already prepared. Usually, a merge is used to prepare a set of the same type of letters, in which, for example, names, addresses, etc. are used as wildcard fields. In our work, we used it to enter prepared functions. The resulting text from Word was simply copied to the MiKTeX editor, with the help of which PDF files containing formulas in the usual form were already being prepared.

3.3. Example of a generator on the topic "Derivatives" in Python

One of the possible ways to implement the generator of tasks by derivatives is the use of recursive functions in the programming of the generator. Recursion is a programming term meaning a function calling itself. This technique can be useful when a task can naturally be divided into several similar, but simpler tasks, which we will use to generate a complex function.

We can conditionally divide all mathematical functions into two large groups:

• unary functions are those functions that have a single argument. For example, trigonometric functions, power, exponential, etc.

• binary – functions whose arguments are two expressions (functions). For example, addition, subtraction, multiplication, division, exponential.

When designing the generator for each of these groups of functions, we developed a corresponding function in the programming language. The function itself had a string type and the result of its operation was a text expression representing the generated task in LaTeX.

Two parameters were passed to these functions as an argument: the variant identifier (IDvariant) and the complexity identifier (IDcomplexity). According to the variant identifier, as we described it at the beginning of the article, the values of variables were determined that determine the choice of a specific mathematical function in LaTeX and (or) the coefficients of this function.

The complexity identifier, in turn, determined the number of levels of function nesting, and when each subsequent function was called, its value decreased by one. Having reached some initial value, the generator stopped calling new functions – it finished working.

Example of unary functions in Python (we omitted some repetitive points by putting a colon in the code):

```
def F1(IDcomplexity,IDvariant):
  if IDcomplexity>0:
    k=str(IDvariant)[IDcomplexity-1]
    if k=='0':
      return '\sin{'+F1(IDcomplexity-1,IDvariant)+'}'
…
    if k in'789':
      return str(random.randint(2,9))+F1(IDcomplexity-1,IDvariant)
  if IDcomplexity==0:
    return 'x'
```

Example of a binary function:

```
def F2(IDcomplexity,IDvariant):
  IDvariant2=int(str(IDvariant)[2:]+ str(IDvariant)[:2])
  if IDcomplexity>1:
    k=str(IDvariant)[IDcomplexity-1]
    if k in '01':
      return F1(IDcomplexity-1,IDvariant)+'+'+F1(IDcomplexity-
1,IDvariant2)
    if k in '234':
      return '\\frac{'+F1(IDcomplexity-1,IDvariant)+'}{'+F1(IDcomplexity-
1,IDvariant2)+'}'
```

As a result of using all these functions, a text string was obtained, written in the required markup format, which, after being loaded into the Moodle SDO and automatically processed by the TECH filter, gave a visual representation of the differential equation.

3.4. Example of a JavaScript task generator

The possibility of placing the generator on the Internet is of great importance [16, 17], so let's consider an example of a generator written in JavaScript.

In the JavaScript program fragment below, two functions are provided: the first generates a random integer in the range specified by the parameters, and the second creates a template for an indefinite integral in TeX and adds an HTML element to the link to which is passed in the parameter of this function. Depending on one set of coefficients, the program will create a template, and using other generated values will substitute numerical coefficients in them, and the KaTeX.js library, in turn, converts command sets into a readable form (we omitted some repetitive moments by putting a colon in the code):

```
function getRandomInt(min, max) {
  return Math.floor(Math.random() * (max - min)) + min;
}
function undefInt(elem) {
  let f1_k1 = getRandomInt(2, 5);
```

```
    let f1_k2 = getRandomInt(2, 9);
…
      let fp1;
      let fp2;
      switch (j1) {
        case 1:
          fp1 = '\\sin^' + f1_k2 + '{' + koef1 + 'x} \\cdot';
          fp2 = '{\\cos{' + koef1 + 'x}} ';
          break;
…
    }
    katex.render('\\int ' + fp1 + ' ' + fp2 + ' dx', elem, {
        throwOnError: false
    });
}
```

The generators discussed above are built on the same principle. We did not consider the generation of tasks for those sections that do not require checking the correctness of the generated data. We have considered the most interesting points from the point of view of generation algorithms: the design of the generator, in which dependencies between the functions used (integrals on the topic) should be used, as well as the generator. Using recursive functions.

4 Discussion of results

One of the possibilities of using our development is integration with learning management systems (LMS), especially in the case of designing a generator in JavaScript (the other examples of technical implementation of generators we have considered require some improvements for this purpose). For example, we can use our generator in conjunction with an open source Moodle - LMS.

The element of generated tasks can harmoniously become part of the electronic training course of Higher Mathematics. Moodle supports the use of resources in a wide range of formats, but we are interested in implementing HTML code, which in turn can include fragments of JavaScript code using the <script> tag. It is enough for us to embed static resources of our development into the built-in WYSIWYG HTML editor so that the generated updated tasks are kept up to date.

Testing, as a possibility of using a generator, can also be carried out by the built-in LMS Moodle tools. In the future, the test results can be easily statistically analyzed, which is an important aspect in training.

The further development of these ideas can be the design of modules that allow the teacher to get an answer to tasks or, more broadly, to get a set of tasks for monitoring students' assimilation of an educational topic, including in the form of a test.

It should be noted that the considered technology can also be used for automated creation of adaptive tests taking into account the level of knowledge of individual students, i.e. the area in which intensive research is also being conducted [18,19].

5 Conclusion

Our proposed approaches to the construction of task generators in higher mathematics have flexible capabilities for their individual configuration and, as can be seen, often (for example, when using the Microsoft Office package) do not even require knowledge of programming

languages. In our examples, we have considered the most difficult cases of generation – the preparation of tasks in integral calculus and differential equations, as well as another approach that uses recursive functions to generate tasks. Meanwhile, in many other branches of mathematics, for example, such as matrices, systems of linear equations and many others, to generate tasks, it was enough for us to simply generate arrays of numbers and substitute them as elements or coefficients.

As a result, we have received a ready-made application that already helps in the educational process, using modern technologies for building the architecture of web applications and digitalization of education. It has the following functionality:

- defining the subject of tasks,
- selecting the number of generated tasks,
- outputting generated jobs to print or to pdf file,
- creating specific templates for exercises,
- calculating the response to the generated task,

as well as opportunities for scaling:

- testing on the generated material,
- generation, recognition and output of QR links,
- gamification elements,
- selection of the difficulty of the generated tasks.

References

1. H. Ebbinghaus, Annals of Neurosciences **20**, 155 (1913). doi:10.5214/ans.0972.7531.200408. ISBN 978-0-7222-2928-6.
2. I.A. Posov, OTO **4, (**2014)
3. V. V. Kruchinin, L. I. Magazinnikov, Yu. V. Morozova, Izvestia TPU **8,** (2006)
4. S. Yudin, Concept, **8** (2016)
5. P. Cristea, R. Tuduce, Automatic generation of exercises for self-testing in adaptive e-learning environments: Exercises on ac circuits. Inter. Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia (Part of WBE), 1126 (WSEAS Publishing, 2011)
6. M. Gangur, Proceedings of the 3rd International Conference on Computer Supported Education, Portugal, SciTePress - Science and Technology Publications, 264 (2011)
7. L. Gladavská, M. Plevný, DIVAI 2014: 10th International Scientific Conference on Distance Learning in Applied Informatics. Prague, Wolters Kluwer, 325 (2014)
8. V. M. Karnaukhov, Computer program for generating control works based on the LaTeX system. Software products and systems **3**, (2010)
9. Ya.Yu. Konovalov, S.K. Sobolev, Mechanical engineering and computer technology **7**, (2016)
10. M. Gangur, Proceedings of the 7th WSEAS/IASME International Conference on Educational Technologies (EDUTE'11), Athens, 129 (WSEAS Press, 2011)
11. Jascha Paris, Mathias Huesing, Burkhard. Corves, Tool²Task – A Software Tool for Automatic Task Generation in Mechanism Theory: 2014-2017, (2019).10.1007/978-3-030-00108-7_15.
12. L.-C. Sung, Y.-C. Lin, M.C. Chen, 7th IEEE International Conference on Advanced Learning Technologies, Proceedings, Niigata, Japan, INSPEC Accession Number: 9868711, (2007). http://dx.doi.org/10.1109/ICALT.2007.56

13. A. Klokov, A. Sorokin, Electronic scientific and Methodological j. of Omsk State University **4(7)**, (2016)
14. G. Goguadze, *ActiveMath – Generation and Reuse of Interactive Exercises using Domain Reasoners and Automated Tutorial Strategies* (Saarland: Saarland University, 2011)
15. C. Zinn, *Program Analysis and Manipulation to Reproduce Learners' Erroneous Reasoning*. In E. Albert (Ed.), LOPSTR 2012. LNCS. 7844, 228 (LNCS, Heidelberg: Springer, 2013).
16. W. Hürst, S. Jung, M. Welte, Proceedings of the 9th international conference on Human computer interaction with mobile devices and services, ACM, pp. 364 (2007)
17. R. Niazi, Q.H. Mahmoud, A poster at the 39th ACM Technical Symposium on Computer Science Education (SIGCSE) (Portland, OR, 2000)
18. J. Kapusta, M. Munk, M. Turčáni, 4th International Conference on Application of Information and Communication Technologies, AICT2010, INSPEC Accession Number: 5611791 (2010). http://dx.doi.org/10.1109/ICAICT.2010.5611791.
19. T. Mine, T. Shoudai, A. Suganuma, Association for the Advancement of Computing in Education (AACE), Chesapeake, VA, 779 (2000)