Automatically generate personalized tasks and sample solutions for the fundamentals of electrical engineering with PGFPlots and CircuiTi*k*Z

Mathias Magdowski

Abstract

All students in our class “Fundamentals of Electrical Engineering” get a personalized task via e-mail, solve it and submit a digitized version of their handwritten solution. Then, students peer review each other (lowering the review effort for the lecturer) with the help of a similarly personalized sample solution.

The procedure is automated via MATLAB and therefore scalable for large numbers of students. In contrast to multiple choice questions or pure numerical solutions, here the approach and the detailed calculations can also be assessed. This article describes generating the tasks and sample solutions in LaTeX with the help of PGFPlots and CircuiTi*k*Z.

1 Motivation

Just like riding a bicycle or playing the piano, you learn the basics of electrical engineering not by watching and listening, but by participating, trying and practicing. In order to prepare our students as well as possible for the corresponding examination after two semesters, we have been offering personalized tasks for handwritten solutions at the Otto von Guericke University in Magdeburg for several years. Such a solution is exam-related and, compared to “online” input in text fields or formula editors, also allows the use of sketches, diagrams and circuit diagrams in the presentation of the approach and calculation path. Personalization of the tasks reduces the danger of copying from an external solution. Nevertheless, the students can, may and should help and advise each other in solving the tasks.

To keep the correction effort for the teachers reasonable, students then correct each other in peer review. This also requires personalized sample solutions, which must be so good and detailed that even someone who has not been able to solve the task correctly can still correct it correctly. Thinking into a foreign solution path also promotes a deeper understanding of the topic of the task. As an extrinsic motivation and incentive for solving the tasks and for mutual assessment, students receive additional points for admission to the exam.

2 Procedure

To prevent the procedure from degenerating into “red tape” for larger groups of students and to ensure that it can be easily automated, the generation of tasks and sample solutions, the submission of solutions, the mutual correction and the allocation of additional points is carried out completely digitally; see Figure 1. The students register in our learning management system Moodle (recognizable by the symbol m). The list of students is fed into a MATLAB program, which automatically generates all personalized tasks and sample solutions using LaTeX. The tasks are then sent to the students by e-mail; the sample solutions are saved into a local folder first.

Next, the students work out their solutions within one week and submit them back to Moodle. Then all solutions can be downloaded in a `zip` file. Another MATLAB program sends all students two foreign solutions as well as the appropriate sample solutions by e-mail for peer review.

The students then must submit their corrections within another week. Again, all corrections and the additional points can be downloaded collectively. A last MATLAB program sends all students their two corrected solutions as well as their own sample solution and notes the achieved points in a list.

MATLAB as a programming environment was chosen by the author because it is quite common in electrical engineering. However, there is already a similar implementation in Python by Olivier Cleynen, who was also working in Magdeburg [1].

3 Implementation in LaTeX

The choice of LaTeX as a document description language was of course not solely made for this special kind of task. Rather, the complete exercise book and the corresponding sample solutions, the catalog of exam questions and the lecture script as a book [3], were already typeset in LaTeX. Especially advantageous are the excellent mathematical notation via the `amsmath` package, the perfect set of physical units via `siunitx`, and the possibility to use PGFPlots or CircuiTi*k*Z based on the Ti*k*Z package to create diagrams and circuit diagrams directly in the LaTeX source code.

When repeatedly creating new exam tasks, the author finally came up with the idea of not changing the numerical values in a diagram by hand, but to use a random number generator for this purpose. If the (pseudo-)random number generator is then started using a personal but known number, e. g. the student number, another number creates a different diagram, but the same number creates the same diagram again. This way, tasks and solutions can be generated multiple times, if something does not work correctly or if small changes have to be made in the text.

doi.org/10.47397/tb/42-2/tb131magdowski-autoee

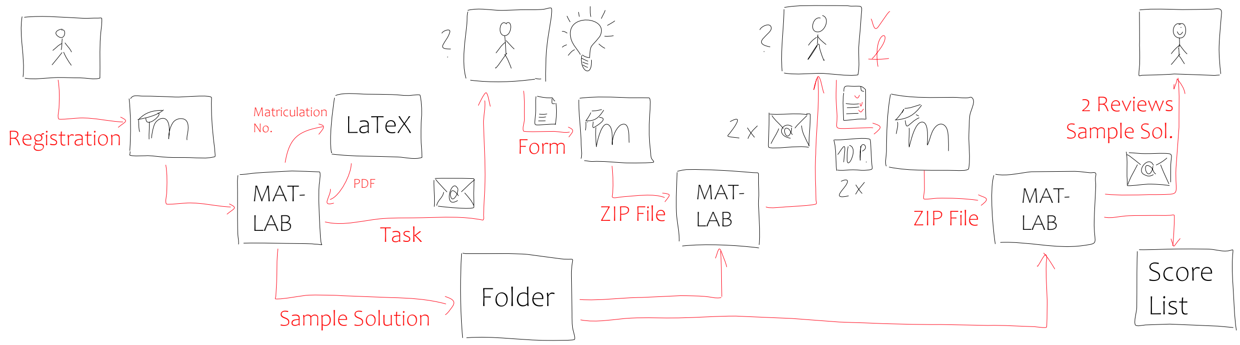

Figure 1: Procedure for the realization of personalizable tasks with anonymous peer review

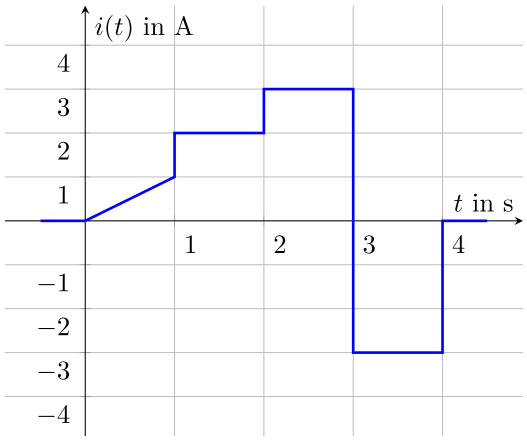

Figure 2: Randomized generated plot in PGFPlots; output of Listing 1

Current in the 1. section (1 point):

$$i(t) = 1\,\mathrm{A\,s^{-1}} \cdot t \tag{1}$$

Charge in the 1. section (1 point):

$$Q(t) = \int_0^t 1\,\mathrm{A\,s^{-1}}t'\,\mathrm{d}t' + 0 \tag{2a}$$

$$= 1\,\mathrm{A\,s^{-1}} \cdot \left[\frac{t'^2}{2}\right]_0^t \tag{2b}$$

$$= 0.5\,\mathrm{A\,s^{-1}}t^2 \tag{2c}$$

Charge at the end of the 1. section (1 point):

$$Q(1\,\mathrm{s}) = 0.5\,\mathrm{A\,s} \tag{3}$$

Figure 3: Algorithmically generated sample solution to the diagram in Figure 2; output of Listing 2

3.1 Random number generation and plot creation directly in LaTeX

A typical task (the same for all students) is the calculation of the time-dependent charge from a graphically given current curve by integrating it section by section over time. All students then receive a personalized time diagram of the current. Listing 1 shows example source code for the generation of the random numbers and the generation of this diagram with the PGFPlots package based on Ti*k*Z. (All listings have been edited slightly for presentation.)

The corresponding output is shown in Figure 2. Each diagram always contains a linear increase (or decrease) of the current in the first time period as well as three further time periods in which the current remains constant (and can also be zero). The greatest difficulty in creating these plots is to use case distinctions to ensure that the current is not the same in two successive time periods. In this case, the number of time segments and thus the computational effort for the students would be reduced.

The LaTeX source for generating the corresponding sample solution is shown in Listing 2. For brevity, it has been shortened to the first time period. The corresponding output is shown in Figure 3. The actual version for students is much more detailed and also deals with typical errors.

One can note that the solution is the same for all students and differs only in the numerical values. This is indeed the case. Nevertheless, the difficulty of this introductory task should not be underestimated, especially for students in the first weeks of the semester when this task is given.

In the case of the independent, handwritten solution, there are many things to pay attention to, in addition to the actual arithmetic solution. These include the reading of the current function from the diagram, the correct mathematical notation, the correct calculation of the units and the drawing of the time-dependent charge. Students must also learn to break down complex tasks into simpler subtasks. The same scheme for generating the diagrams and the appropriate sample solution is also used in a

Listing 1: LaTeX source code to create randomized PGFPlots diagrams; output is shown in Figure 2

```
\documentclass{standalone}
\usepackage{pgfplots,siunitx}
\begin{document}
% Set random number generator to matriculation number
\pgfmathsetseed{123456}
% Current at time 1 s (in A, can also still be zero, but should not be)
\pgfmathrandominteger{\stromeinsrandom}{-4}{4}
% if current is zero, set to 1 A
\pgfmathsetmacro{\stromeins}{ifthenelse(\stromeinsrandom==0,1,\stromeinsrandom)}
% Current in the period from 1 s to 2 s (in A, can also be zero)
\pgfmathrandominteger{\stromzwei}{-4}{4}
% Current in the period from 2 s to 3 s (in A, can also be zero)
\pgfmathrandominteger{\stromdreirandom}{-4}{4}
% if the current is equal to the value from the previous period, invert the sign
\pgfmathsetmacro{\stromdrei}{ifthenelse(\stromzwei==\stromdreirandom,
                                        -\stromdreirandom,\stromdreirandom)}
% if both currents are zero, set new current to 1 A
\pgfmathsetmacro{\stromdrei}{ifthenelse(abs(\stromzwei)+abs(\stromdrei)==0,1,\stromdrei)}
% Current in the period from 3 s to 4 s (in A, can also be zero)
\pgfmathrandominteger{\stromvierrandom}{-4}{4}
% if both currents are zero, set new current to 1 A
\pgfmathsetmacro{\stromvier}{ifthenelse(abs(\stromzwei)+abs(\stromvierrandom)==0,
                                        1,\stromvierrandom)}
% if both currents are zero, set new current to 1 A
\pgfmathsetmacro{\stromvier}{ifthenelse(abs(\stromdrei)+abs(\stromvier)==0,1,\stromvier)}
% if the current is equal to the value from the previous period, invert the sign
\pgfmathsetmacro{\stromvier}{ifthenelse(\stromdrei==\stromvier,-\stromvier,\stromvier)}
\begin{tikzpicture}
        \begin{axis}[
                xlabel={$t$ in \si{\second}}, ylabel={$i(t)$ in \si{\ampere}},
                xmin=-0.9,xmax=4.9,ymin=-4.9,ymax=4.9,
                xtick={1,2,3,4},ytick={-4,-3,-2,-1,1,2,3,4},
                xticklabel style={below right},
                yticklabel style={below left},
                axis x line=middle,axis y line=center,
                xmajorgrids,ymajorgrids,]
                \addplot+[mark=none,line width=1pt] coordinates {
                        (-0.5,0)
                        (0,0)
                        (1,\stromeins) (1,\stromzwei)
                        (2,\stromzwei) (2,\stromdrei)
                        (3,\stromdrei) (3,\stromvier)
                        (4,\stromvier) (4,0)
                        (4.5,0)};
        \end{axis}
\end{tikzpicture}\end{document}
```

task where students have to determine the mean and effective value of periodic time functions.

3.2 Random number generation and LaTeX source code creation for circuit diagrams in MATLAB

Further typical tasks relate to the core area of the fundamentals of electrical engineering, circuit calculation or linear network analysis. The CircuiTi*k*Z package, which is also based on Ti*k*Z, can be used to represent electrical networks. With this package, basic elements such as resistors or voltage and current sources can be placed along paths. In addition, the elements and nodes can be labeled, current and voltage arrows can be drawn in, etc. The elements are placed according to a Ti*k*Z coordinate system.

The generation of different circuit diagrams for practicing the nodal analysis, which all have approximately the same complexity and the same computational effort, can for example be based on a fixed

Listing 2: Source code for the algorithmic generation of the matching sample solution to the diagram in Figure 2; for output see Figure 3

```
% Truncate decimal places for an integer number
\pgfmathdeclarefunction{trimzero}{1}
  {\pgfmathparse{ifthenelse(#1==round(#1),
                            int(#1),#1)}}
% Convert current (in A) to integers
\pgfmathsetmacro{\stromeins}{int(\stromeins)}
% Charge at the end of the 1st section (in As)
\pgfmathsetmacro{\ladungeins}{trimzero(\stromeins/2)}
% Differential operator (small upright d)
\newcommand*{\diff}{\mathop{}\!\mathrm{d}}

Current in the 1. section (1 point):
\begin{equation}
  i(t) = \SI{\stromeins}{\ampere\per\second} \cdot t
\end{equation}

Charge in the 1. section (1 point):
\begin{subequations}\begin{align}
  Q(t) &= \int \limits_0^t
                 \SI{\stromeins}{\ampere\per\second}
                 t’ \diff t’ + 0 \\
  &= \SI{\stromeins}{\ampere\per\second}
     \cdot \left[ \frac{t’^2}{2} \right]_0^t \\
  &= \SI{\ladungeins}{\ampere\per\second} t^2
\end{align}\end{subequations}

Charge at the end of the 1. section (1 point):
\begin{equation}
  Q(\SI{1}{\second}) = \SI{\ladungeins}
                          {\ampere\second}
\end{equation}
```

number of three nodes in the network, which are then connected to each other with different branches. With three nodes 1, 2 and 3 (plus the reference node 0), there are six possible branches ($0 \leftrightarrow 1$, $0 \leftrightarrow 2$, $0 \leftrightarrow 3$, $1 \leftrightarrow 2$, $1 \leftrightarrow 3$ and $2 \leftrightarrow 3$). Possible options for the branches are an open circuit, a resistor, a current source or a parallel connection of a resistor and a current source when calculating with DC currents and voltages. So there are $4^6 = 4096$ different possibilities.

Like the random values in the previous task for charge and current, the branches cannot be chosen completely arbitrarily, because it must be ensured that the generated network is actually computable. Therefore it has to be checked if there is at least one source in the network. Furthermore it has to be checked if all node sets can be fulfilled. For example, the network could be overdetermined by too many current sources or too many open-circuited branches. In this case, another parallel resistor must be added to a current source, or a current source or an open circuit must be removed and replaced by a resistor, respectively. To check whether a network is computable, the algorithm can calculate, for instance, the determinant of the nodal conductance matrix and check for inequality with zero.

Listing 3: Source code (created automatically from MATLAB) for a circuit diagram using CircuiTi*k*Z for practicing nodal analysis; for output see Figure 4

```
\documentclass{standalone}
\usepackage{amsmath}
\newcommand{\ind}[1]{\mathrm{#1}}
\usepackage{circuitikz}
\begin{document}\begin{tikzpicture}[scale=1.3]
\draw (2,0) to[short] (0,0);
\draw (0,0) to[I, i^>=$I_{\ind{q}1}$] (0,2);
\draw (0,0) to[short, *-] (-2,0)
  to[R, l=$R_{1}$] (-2,2) to[short, -*] (0,2);
\draw (2,0) to[short] (4,0);
\draw (4,0) to[I, i_>=$I_{\ind{q}2}$] (4,2);
\draw (2,0) to[R, l^=$R_{2}$] (2,2);
\draw (0,2) to[short] (0,4);
\draw (0,4) to[I, i^<=$I_{\ind{q}3}$] (4,4);
\draw (4,4) to[short] (4,2);
\draw (0,2) to[I, i<^=$I_{\ind{q}4}$] (2,2);
\draw (4,2) to[R, l_=$R_{3}$] (2,2);
\node[circ] at(2,0) {};\node[below] at (2,0) {0};
\node[circ] at(0,2) {};\node[above left] at(0,2) {1};
\node[circ] at(4,2) {};\node[above right] at(4,2) {2};
\node[circ] at(2,2) {};\node[above] at(2,2) {3};
\end{tikzpicture}\end{document}
```

In principle, it would be possible to perform all these queries and calculations directly in LaTeX. Instead, the author has decided to outsource the randomization of the tasks for nodal analysis to MATLAB due to the programming effort involved and to generate the corresponding LaTeX source code for the tasks as well as sample solutions from there. To export the LaTeX source from MATLAB, the simple `fprintf` command is used. The `dos` command can also be used to call `pdflatex` directly from MATLAB in order to create PDF files and send them to the students using the `sendmail` function [2].

Listing 3 (corresponding output in Figure 4) shows an example of LaTeX source code automatically generated in MATLAB for a circuit to practice node voltage analysis. The MATLAB program generating the LaTeX source runs in a loop over all branches of the network and outputs the corresponding source code depending on the circuit element. The $0 \leftrightarrow 1$, $0 \leftrightarrow 2$ and $1 \leftrightarrow 2$ branches require a pre-path or post-path from the node to the circuit element and/or back. Afterwards, a loop runs over all nodes, checks if more than two branches with circuit elements are

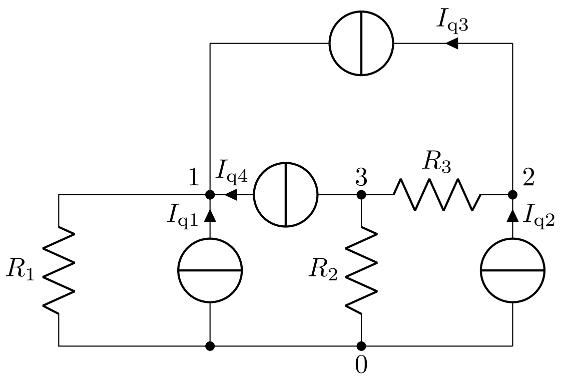

Figure 4: Randomized generated circuit diagram in CircuiTi*k*Z; output of Listing 3

Listing 4: Source code (created automatically from MATLAB) for the system of equations to calculate the circuit from Figure 4 using nodal analysis; for output see Figure 5

```
\begin{equation*}
\begin{bmatrix}
G_{1} & 0 & 0 \\
0 & G_{3} & -G_{3} \\ 0 & -G_{3} & G_{2} + G_{3}
\end{bmatrix} \cdot \begin{bmatrix}
U_{\ind{Kn}1} \\ U_{\ind{Kn}2} \\ U_{\ind{Kn}3}
\end{bmatrix} = \begin{bmatrix}
I_{\ind{q}1} + I_{\ind{q}3} + I_{\ind{q}4} \\
I_{\ind{q}2} - I_{\ind{q}3} \\ - I_{\ind{q}4}
\end{bmatrix}
\end{equation*}
```

$$\begin{bmatrix} G_1 & 0 & 0 \\ 0 & G_3 & -G_3 \\ 0 & -G_3 & G_2+G_3 \end{bmatrix} \cdot \begin{bmatrix} U_{\mathrm{Kn1}} \\ U_{\mathrm{Kn2}} \\ U_{\mathrm{Kn3}} \end{bmatrix} = \begin{bmatrix} I_{\mathrm{q1}}+I_{\mathrm{q3}}+I_{\mathrm{q4}} \\ I_{\mathrm{q2}}-I_{\mathrm{q3}} \\ -I_{\mathrm{q4}} \end{bmatrix}$$

Figure 5: Algorithmically generated sample solution to calculate the network in Figure 4 using nodal analysis; output of Listing 4

connected to the node and, in this case, draws a solder point at the node.

The LaTeX source code for the corresponding sample solution is at least partially generated in MATLAB. An excerpt of the system of equations for the calculation of the network is shown in Listing 4 (corresponding output in Figure 5).

Further task types for nodal analysis with complex admittances in the frequency domain, for the equivalent resistance and impedance of a circuit, for the voltage divider rule and the phasor diagram can be found at `https://bit.ly/12PeerTasks`.

4 Conclusion & outlook

With a little conceptualization and programming effort (the author previously needed about 15 h to 20 h per task), randomized or personalized tasks can be created directly in LaTeX, or via source code generation in MATLAB, to practice basic computing procedures in the fundamentals of electrical engineering. With some creativity, the procedure could be transferred to other disciplines in the engineering sciences.

The concept is already tested in practice and well accepted by students. As an extension, additional task types are conceivable; for example, the current divider rule or stepwise calculation of current and voltage in a branched network. In addition, tasks with varying degrees of difficulty for internal differentiation in heterogeneous student groups would also be possible. Up to now, all students have received a task with similar difficulty and comparable calculation effort. In the example task on charge and current, the number of time steps could be increased or more linear, or even quadratic or exponential time functions could be inserted instead of constant charge functions. In the example task for the nodal analysis, networks with more nodes and thus more potential branches could be generated, whose computation requires a larger system of equations with more equations and more unknowns.

From a programming point of view, it would be interesting to implement more of the algorithms for task and solution generation in LaTeX instead of generating the source code externally in MATLAB. For the overall concept, it would be desirable if the generation and provision of the tasks for the students, the assignment and processing of the mutual assessments and the accounting of the achieved points would completely work in a learning management system like Moodle.

References

[1] O. Cleynen, G. Santa-Maria, et al. Peer-graded individualized student homework in a single-instructor undergraduate engineering course. *Research in Learning Technology* 28, May 2020. `10.25304/rlt.v28.2339`

[2] D. Gleich. Get Matlab to email you when it's done running!, Feb. 2014. `https://dgleich.wordpress.com/2014/02/27/`

[3] J.B. Nitsch, U. Knauff, M. Magdowski. *Einführung in die Elektrotechnik.* Shaker-Verlag, Aachen, 2nd ed., July 2011.

◇ Mathias Magdowski
Otto von Guericke University,
Magdeburg, Germany
`mathias.magdowski (at) ovgu dot de`