# TEACHING ELECTRICAL ENGINEERING USING WOLFRAM LANGUAGE

Miroslav Lutovac[1],
Vladimir Mladenović[2]

[1]Singidunum University,
32 Danijelova Street, Belgrade, Serbia
[2]University of Kragujevac,
Faculty of Technical Sciences,
Čačak, Serbia

Abstract:
The undergraduate engineering degree programs usually require good mathematical knowledge because the electrical courses, such as Electrical engineering and Electronics, heavily rely on school and advanced mathematics. On the other hand, in order to start with exercises and experiments, as soon as possible, the appropriate tools and support will benefit for success. This paper presents a new approach that describes the usage of Wolfram language. It is shown that basic expressions and circuit analysis procedures can be transformed into the program code for immediate visualization of basic formulas and properties, and expand the logical functions into the form that can be realized with appropriate logic circuits.

Key words:
software tools, algorithms, electronics.

Acknowledgment:
This work was supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia under Grant TR 32023.

## 1. INTRODUCTION

Some teachers with excellent mathematical knowledge consider electrical engineering nothing more than applied mathematics. Therefore, in the traditional education, the first-year university students, sometimes called freshers or freshman, are usually focused on the mathematics and the basic electrical engineering with numerous mathematical derivations and mathematical proofs of theorems. Nowadays, many students are not willing to work hard before starting with the application of that knowledge (such as the electrical courses), especially if software engineering is in the center of attention. On the other hand, it is important to avoid teaching in the form *Electrical engineers for dummies*.

The purpose of this paper is to motivate students to attend electrical engineering courses and to use the appropriate software environments for overcoming weak knowledge of mathematics. Nowadays, we will not use slide rules (slipstick) or analog computer to perform mathematical computations, although they were very popular before appearance of pocket digital calculator. Following the same philosophy, modern engineers are not less good engineers if they are using computer algebra systems for deriving properties or proving mathematical theorems.

Correspondence:
Miroslav Lutovac

e-mail:
mlutovac@singidunum.ac.rs



DOI: 10.15308/Sinteza-2016-221-225

In order to provide thorough understanding of fundamental concepts in electrical engineering, a great number of computer distance assisted learning programs have been released. They can be used on personal computers or tablets and smart phones.

For advanced simulations and complex packages, workstations and cloud solutions can be used. Usually, the course materials are organized in modular syllabus structures. The interactivity, real-time calculations and self-assessment are required features for delivery units (Froyd *et al.*, 2012). For pedagogical reasons, the students of the same group can teach other less successful students, and this way preparing them for team working. Some students will have better mathematical knowledge, while other students will have better understanding of electrical principles and devices. As a team, they will encourage some students to solve problems that are unfamiliar for the rest of the team.

The most-favored pedagogical model for teaching electrical engineering is the project-based learning; as fundamental methods of solving problems using trial and error, generate and test, or guess and check, which are based on the software tools and appropriate applications (Guzdial & DiSalvo, 2013), students can learn without damaging measurement devices and electrical components.

During the process of designing some meaningful systems, students of electrical engineering will enhance design thinking skills. Some students will still have problems with their understanding of mathematics, and the purpose of this approach is to overcome frustration and disappointment.

We expect that some students with such unsatisfactory knowledge will concern about their interpretation of mathematics in learning electrical engineering. Our assumption is that students will worry how to transform the textbook mathematical expressions into the real world electronic systems and practical exercises. This means that we should focused on transformations from the real world into the mathematical representations, derive properties in the mathematical space, and finally, transform the mathematical expressions into the real word meaningful engineering context. To do that, we will explain the role of symbols that represent some physical quantities, the conventions for interpreting the symbols in a proper domain, and finally present how the equations can be interpreted. Experience with teaching electonics was presented in several papers (Lutovac & Mladenović, 2015a, 2015b; Lutovac *et al.*, 2015).

## 2. CALCULATION AND VISUALIZATION OF ELECTRICAL QUANTITIES

### *Calculation and Visualization of Electric Field*

As the first example, let us consider Coulomb's law that describes force interacting between two static electrically charged particles, $Q_1$ and $Q_2$. The vector form of the electrostatic force $\vec{F}_{12}$ on the charge $Q_2$ in the vicinity of the charge $Q_1$ is as follows (assuming that $\vec{r}_{012}$ is the unit vector and $k$ is the constant):

$$\vec{F}_{12} = k\frac{Q_1Q_2}{r^2}\frac{\vec{r}_{12}}{r_{12}} = k\frac{Q_1Q_2}{r^2}\vec{r}_{012} \qquad (1)$$

By using the law of superposition, the force $\vec{F}_p$ on the charge $Q_p$, due to a system of $n$ discrete charges $\{Q_1, Q_2, Q_3,..., Q_n\}$ is:

$$\vec{F}_p = \vec{F}_{1p} + \vec{F}_{2p} + \vec{F}_{3p} + ... + \vec{F}_{np} \qquad (2)$$

The electric field $\vec{E}$ is also a vector that exist in space at the point where is a small stationary test particle of unit charge $Q_p$:

$$\vec{E} = \frac{\vec{F}_p}{Q_p} \qquad (3)$$

When two discrete charges are considered, the visualization of forces and electric field is also very simple. Whit more charges with different values, such as illustrated in Figure 1, plotting can be a serious problem.

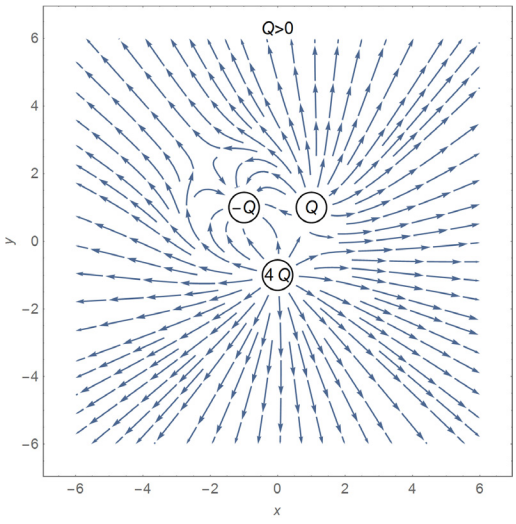


Figure 1. Illustration of the electric field surrounding three charges $\{Q_1=Q, Q_2=-Q, Q_3=4Q\}$, where $Q>0$.

The Wolfram language (Wolfram, 2015) can be used for defining the electric field in terms of electric charge and the position of that charge in the space (in this case it is a function of *x* and *y* as Cartesian coordinate system in the plain). To enter the knowledge into our environment, the first step is to define a function that will return some values for some known input quantities. We can use := to assigns some procedure described at the right side of = so the result will not be evaluated at the moment of writing the code. That is, the procedure at the right side will remain unevaluated, this is just a knowledge of something. In this example, knowledge is the expression describing electric field. On the left side of := we define the name of the function (Ee) and list of expected arguments (q, x, y, x0, y0). The symbol with associated underscore (q_) means that it is a pattern object that can stand for any Wolfram Language expression that will be used as local variable on the right side (q). This knowledge of the electric field at the point {x, y} in a plain is a pair of two values that is a function of charge q that exists at the position {x0, y0}, see Figure 2:

```
Ee[q_, x_, y_, x0_, y0_] :=
 {q / (4 π ϵ0 ((Abs[x - x0] + ϵx)^2 + (Abs[y - y0] + ϵy)^2))
   (x - x0) / Sqrt[(Abs[x - x0] + ϵx)^2 + (Abs[y - y0] + ϵy)^2]),
  q / (4 π ϵ0 ((Abs[x - x0] + ϵx)^2 + (Abs[y - y0] + ϵy)^2))
   (y - y0) / Sqrt[(Abs[x - x0] + ϵx)^2 + (Abs[y - y0] + ϵy)^2])}
```

Figure 2. Deffining the electric field produced by single charge.

Some values on the right side of := are global variables and constants, such as $\varepsilon_0$, $\varepsilon_x$, and $\varepsilon_y$, that we can define before or after entering the knowledge, as shown in Figure 3:

```
qe = 1.602 × 10^-19; (* C *)
ϵ0 = 8.8542 × 10^-12; (* F/m *)
ϵx = 1 × 10^-12; ϵy = 1 × 10^-12;
ne = 10^7;
q = qe ne;
```

Figure 3. Definning the constants.

In the next part we define other specific variables, such as elementary charge, and three charges with their positions in the plain, see Figure 4. Finally, we calculate electric filed at still unknown position {x, y}:

```
Q1 = 1; x1 = 1; y1 = 1;
Q2 = -1; x2 = -1; y2 = 1;
Q3 = 4 ; x3 = 0; y3 = -1;
E1 = Ee[Q1 q, x, y, x1, y1];
E2 = Ee[Q2 q, x, y, x2, y2];
E3 = Ee[Q3 q, x, y, x3, y3];
```

Figure 4. Setting the particular values and commputing the electric field of each charge.

By using the superposition, we are calculating electric field in Cartesian coordinate system.

```
StreamPlot[{Re[E1 + E2 + E3], Im[E1 + E2 + E3]},
```

Figure 5. Command Streamplot for plotting the electric field.

The reason for specifying $\varepsilon_x$ and $\varepsilon_y$ is that to avoid infinitive values of the field in the center of charges. The command StreamPlot generates a stream plot of the vector field, Figure 5.

*Computation of Electrostatic Equipotentials Between Three Electrically Charged Spheres*

All variables, already used for plotting electrical field, can be used for illustrating other characteristics. We do not need the vector presentation; therefore, the required knowledge can be as in Figure 6.

```
Enorm = 1000;
Ed[q_, x_, y_, x0_, y0_] :=
 q / (4 π ϵ0 ((Abs[x - x0] + ϵx)^2 + (Abs[y - y0] + ϵy)^2)) Enorm

Ed1 = Ed[Q1 q, x, y, x1, y1];
Ed2 = Ed[Q2 q, x, y, x2, y2];
Ed3 = Ed[Q3 q, x, y, x3, y3];
```

Figure 6. Defining the electric field of single charged spheres for plotting electrostatic equipotentials.

By using the superposition, we are calculating electrostatic equipotentials in Cartesian coordinate system, as in Figure 7.

```
ContourPlot[{Ed1 + Ed2 + Ed3}, {x, -4.5, 5}, {y, -5, 4.5},
```

Fig. 7. Command ContourPlot for plotting the electrostatic equipotentials.

The command ContourPlot generates a contour plot, Figure 8.

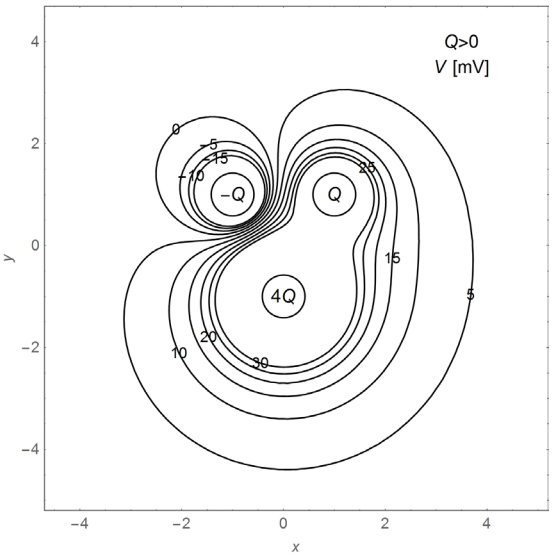


Figure 8. Illustration of electrostatic equipotentials of three charges $\{Q_1=Q, Q_2=-Q, Q_3=4Q\}$, where $Q>0$.

## 3. DESIGN OF DIGITAL INTEGRATED CIRCUITS

Digital circuits are designed using logic gates. Logic gate are used for performing boolean logic for creating combinational logic.

Integrated circuits consist of a great number of logic gates of just a few different logic types, so that the designer has a task to interconnect logic gates. Integrated circuits are designed using automation software to perform specific type of functions. Instead of relying on a single software solution, sometimes designers would prefer to take a full control on the design process, especially if some specific conditions are considered for embedded hardware.

Wolfram language can be efficiently used for performing this job. Suppose that we have on disposal a specific type of logic circuits, all of them in available forms. For logic optimization, specific commands can be used. `LogicalExpand` puts logical expressions into a standard disjunctive normal form (DNF), consisting of an OR of ANDs. Other forms are as follows:

| | |
|---|---|
| "DNF","SOP" | disjunctive normal form, sum of products, |
| "CNF","POS" | conjunctive normal form, product of sums, |
| "ANF" | algebraic normal form |
| "NOR" | two-level NOR |
| "NAND" | two-level NAND |
| "AND" | two-level AND or Not |
| "OR" | two-level OR and Not |

Two-level NOR and NAND logic circuits can be used as Not when both logic inputs are connected to the same signal.

*Example of Optimisation*

As an example, we will consider the logic function $x$

$$x = (a\wedge b\wedge\neg c)\vee(a\wedge\neg b\wedge c)\vee(a\wedge\neg c\wedge d)\vee(\neg a\wedge b\wedge c)\vee(b\wedge c\wedge\neg d)\vee(b\wedge\neg c\wedge d)\vee(\neg b\wedge c\wedge d)$$

One possible solution for implementation can be by using two-input NOR gates. The command **BooleanConvert** will perform that transformation:

```
y = BooleanConvert[x, "NOR"]
y // TraditionalForm
```

The results of using the Wolfram language notation is:

$$(!a\bar{\vee}!b\bar{\vee}!c\bar{\vee}!d)\bar{\vee}(a\bar{\vee}b\bar{\vee}c)\bar{\vee}(a\bar{\vee}b\bar{\vee}d)\bar{\vee}(a\bar{\vee}c\bar{\vee}d)\bar{\vee}(b\bar{\vee}c\bar{\vee}d)$$

Traditional format for the mathematical representation is:

$$(\neg a\bar{\vee}\neg b\bar{\vee}\neg c\bar{\vee}\neg d)\bar{\vee}(a\bar{\vee}b\bar{\vee}c)\bar{\vee}(a\bar{\vee}b\bar{\vee}d)\bar{\vee}(a\bar{\vee}c\bar{\vee}d)\bar{\vee}(b\bar{\vee}c\bar{\vee}d)$$

We can use two-input NAND logic circuits:

```
y = BooleanConvert[x, "NAND"]
y // TraditionalForm
```

The result of using Wolfram language notation is:

$$(a\bar{\wedge}b\bar{\wedge}!c)\bar{\wedge}(a\bar{\wedge}!b\bar{\wedge}c)\bar{\wedge}(a\bar{\wedge}!c\bar{\wedge}d)\bar{\wedge}(!a\bar{\wedge}b\bar{\wedge}c)\bar{\wedge}(b\bar{\wedge}c\bar{\wedge}!d)\bar{\wedge}(b\bar{\wedge}!c\bar{\wedge}d)\bar{\wedge}(!b\bar{\wedge}c\bar{\wedge}d)$$

Traditional format for the mathematical representation is:

$$(a\bar{\wedge}b\bar{\wedge}\neg c)\bar{\wedge}(a\bar{\wedge}\neg b\bar{\wedge}c)\bar{\wedge}(a\bar{\wedge}\neg c\bar{\wedge}d)\bar{\wedge}(\neg a\bar{\wedge}b\bar{\wedge}c)\bar{\wedge}(b\bar{\wedge}c\bar{\wedge}\neg d)\bar{\wedge}(b\bar{\wedge}\neg c\bar{\wedge}d)\bar{\wedge}(\neg b\bar{\wedge}c\bar{\wedge}d)$$

We can use two-input AND and OR logic circuits, and not gates for the realization of disjunctive normal form:

```
y = BooleanConvert[x, " DNF"]
y // TraditionalForm
```

Result using Wolfram language notation is:

$$(a\bar{\wedge}\ b\bar{\wedge}!c)||(a\bar{\wedge}!b\bar{\wedge}c)||(a\bar{\wedge}!c\bar{\wedge}d)||(!a\bar{\wedge}b\bar{\wedge}c)||(b\bar{\wedge}c\bar{\wedge}!d)||(b\bar{\wedge}!c\bar{\wedge}d)||(!b\bar{\wedge}c\bar{\wedge}d)$$

Traditional format for the mathematical representation is:

$$(a\wedge b\wedge\neg c)\vee(a\wedge\neg b\wedge c)\vee(a\wedge\neg c\wedge d)\vee(\neg a\wedge b\wedge c)\vee(b\wedge c\wedge\neg d)\vee(b\wedge\neg c\wedge d)\vee(\neg b\wedge c\wedge d)$$

We can use two-input AND and OR logic circuits, and not gates for the realization of conjunctive normal form:

```
y = BooleanConvert[x, " CNF"]
y // TraditionalForm
```

The result of using Wolfram language notation is:

$$(!a||!b||!c||!d)\&\&(a||b||c)\&\&(a||b||d)\&\&(a||c||d)\&\&(b||c||d)$$

Traditional format for the mathematical representation is:

$$(\neg a\vee\neg b\vee\neg c\vee\neg d)\wedge(a\vee b\vee c)\wedge(a\vee b\vee d)\wedge(a\vee c\vee d)\wedge(b\vee c\vee d)$$

We can use two-input AND logic circuits and not gates:

```
y = BooleanConvert[x, "AND"]
y // TraditionalForm
```

The result of using Wolfram language notation is:

$$!(a\&\&b\&\&c\&\&d)\&\&!(!a\&\&!b\&\&!c)\&\&!(!a\&\&!b\&\&!d)\&\&!(!a\&\&!c\&\&!d)\&\&!(!b\&\&!c\&\&!d)$$

Traditional format for the mathematical representation is:

$$\neg(a\wedge b\wedge c\wedge d)\wedge\neg(\neg a\wedge\neg b\wedge\neg c)\wedge\neg(\neg a\wedge\neg b\wedge\neg d)\wedge\neg(\neg a\wedge\neg c\wedge\neg d)\wedge\neg(\neg b\wedge\neg c\wedge\neg d)$$

We can use two-input OR and Not gates:

```
y = BooleanConvert[x, "OR"]
y // TraditionalForm
```

The result of using Wolfram language notation is:

$$!(!a||!b||c)||!(!a||b||!c)||!(!a||c||!d)||!(a||!b||!c)||!(!b||!c||d)||!(!b||c||!d)||!(b||!c||!d)$$

Traditional format for the mathematical representation is:

$$\neg(\neg a\vee\neg b\vee c)\vee\neg(\neg a\vee b\vee\neg c)\vee\neg(\neg a\vee c\vee\neg d)\vee\neg(a\vee\neg b\vee\neg c)\vee\neg(\neg b\vee\neg c\vee d)\vee\neg(\neg b\vee c\vee\neg d)\vee\neg(b\vee\neg c\vee\neg d)$$

| Form | ANF | DNF | CNF | AND | OR | NAND | NOR |
|---|---|---|---|---|---|---|---|
| #circuits | 27 | 27 | 19 | 32 | 21 | 27 | 19 |

Table I. Required two-input logic circuits and Not gates

Calculating the count of the number of two-input logic circuits and one-input Not logic gates the most efficient realization can be selected, as it is shown in Table I.

Design and analysis of electric circuits using computer algebra systems (CAS) is presented in (Lutovac *et al.* 2001; Lutovac, 2015), where the same environment is used for drawing schematics and deriving properties of circuits.

For simplifying the design and analysis using CAS, software is released for analog and digital systems (Lutovac & Tosic, 2015).

## 4. SUMMARY

A new approach to teaching electrical engineering at Singidunum University is based on software tools for knowledge presentation and visualization and students homework. The main issue is to understand the theory and to avoid complex mathematical proving and derivations. The students are encouraged to use free open source software environments and vendor canned software. Computer algebra systems, such as that based on Wolfram language, is used for preparing all figures in the main textbook, and all examples are solved using symbolic processing.

## REFERENCES

Froyd, J.E., Wankat, P.C., & Smith, K.A. (2012). Five major shifts in 100 years of engineering education. *Proceedings of the IEEE*, 100 (Special Centennial Issue), 1344-1360.

Guzdial, M., & DiSalvo, B., (2013). Computing Education: Beyond the Classroom. *Computer*, 46(9), 30-31.

Hambley, R. (2011). *Electrical Engineering, Principles and Applications.* Upper Saddle River: Prentice Hall.

Lutovac, M. (2015). *Electrical Engineering.* (in Serbian) Belgrade, Serbia: Singidunum University.

Lutovac, M., & Mladenović, V. (2015). Contemporary Electronics with LTSPICE and Mathematica. In Sinthesis - International Scientific Conference of IT and Business-Related Research, April 2015 (pp. 134 – 138). Belgrade, Serbia: Singidunum University.

Lutovac, M., & Mladenović, V. (2015). Teaching Contemporary Electronics Using Wolfram Language. In International Conference on Electrical, Electronic and Computing Engineering – IcETRAN, 8-11 June 2015, (pp. EKI1.2.1 – EKI1.2.4). Silver Lake, Serbia: ETRAN Society.

Lutovac, M., Spalević, P., & Arsić, N. (2015). Raspberry Pi, Mathematica, and Electrical Engineering Education, Computational Technologies, In The Bulletin of KazNU, 3(86), 2015. 24-27 September 2015, Vol 20, 2015, (pp. 139 – 151), Almaty, Kazakhstan: Al-Farabi Kazakh National University, Kazakhstan.

Lutovac, D., Tosic, V. &Evans L. (2001). *Filter Design for Signal Processing Using MATLAB and Mathematica.* Upper Saddle River, Prentice Hall.

Lutovac, M., & Tosic, V. (2015). *SchematicSolver 2.3 for Mathematica 9.* Belgrade, Serbia: Lutovac Publisher.

Wolfram, S. (2015). An Elementary Introduction to the Wolfram Language. Champaign, IL: Wolfram Media.