Informatics in Education, 2020, Vol. 19, No. 2, 239–256
© 2020 *Vilnius University*
DOI: 10.15388/infedu.2020.12

# Arduino and Numerical Mathematics

Đorđe HERCEG[1], Dejana HERCEG[2]
[1]*Department of Mathematics and Informaics, Faculty of Science, University of Novi Sad*
*Novi Sad, Serbia*
[2]*Department of Power, Electronic and Telecommunication Engineering*
*Faculty of Technical Sciences, University of Novi Sad*
*Novi Sad, Serbia*
*e-mail:herceg@dmi.uns.ac.rs, vuletic@uns.ac.rs*

Received: November 2019

**Abstract.** Connecting theory and practice in teaching is sometimes difficult, as it requires expensive or delicate equipment, thus limiting the teacher to giving demonstrations in which students are passive participants. Numerical mathematics, as an applied discipline, should be taught on real world examples. By using inexpensive Arduino hardware, we can create simple experiments that are easily reproduced by students. Furthermore, the experiments generate tangible data, which can be processed numerically. The choice of the software used for numerical processing is also an important issue. We present several exercises in numerical mathematics that are based on experiments in electrical engineering with Arduino, and show how to turn them into motivational examples. We also present our experiences in teaching using the developed exercises, as well as some important points and conclusions, which stem from discussions with the participating students and teachers.

**Keywords**: interactive learning environments, numerical mathematics, Arduino, education, teaching.

## 1. Introduction

Numerical mathematics is studied in some middle schools and in almost all mathematics courses at undergraduate level studies. Interpolation is introduced by means of Lagrange polynomial and Newton polynomials. The concept of definite integral is introduced as the Riemann integral, which is defined in terms of Riemann sums and its geometric interpretation. Numerical differentiation is devised from Taylor series and explained geometrically as the slope of the secant. While these topics are important, demonstrations and exercises are often made in class using some arbitrary data, without obvious connection to the real world. By using inexpensive Arduino hardware, a teacher can obtain real world data for students to work on, as well as demonstrate the connection between theory and application. We present several examples that were developed with the aim to motivate the students and raise their interest in solving real world problems using the knowledge and techniques learned in class. Our examples are based on Arduino Uno R3, GeoGebra, *Mathematica*, C++ and C# development environments. This work is a continuation of our previous work on teaching of Numerical mathematics using computers (Herceg and Herceg, 2008; Herceg and Herceg, 2009; Herceg and Herceg, 2010).

The paper is organized as follows. Section 2 describes motivation and related work. Sections 3, 4 and 5 introduce interpolation, numerical integration and numerical differentiation through a brief theoretical introduction and with examples developed with Arduino, some programming, GeoGebra and *Mathematica*. A real world problem is solved for each topic and the connection between theory and practice is established. Section 6 contains an overview of feedback received from the participants in our courses. The results of a pedagogical experiment are in Section 7. Concluding remarks are given in Section 8.

## 2. Motivation and Related Work

Interpolation, numerical integration and numerical differentiation are topics in the Numerical mathematics course, taught at the Faculty of Science, University of Novi Sad. The ‘classical’ approach to teaching consists of oral lectures supported with slides, followed by practical exercises during which the students solve computational problems on paper, using the computer software (*Mathematica* and/or GeoGebra) mainly as a programmable calculator.

We sought to improve our teaching materials, classroom activities and exercises by following the ideas and principles explored in the contemporary literature. Having in mind *the new demands occurring in educational systems (and) in order to prepare students for future professions where technology offers enormous opportunities for teaching and learning,* we chose to *rethink our educational paradigm and strategies* (Drijvers *et al.*, 2016).

In Cheng *et al.* (2016) it is argued that majority of teaching is based on the ‘teaching by telling’ approach which centers on mathematical derivation and ignores the natural connection between theory and application. They propose introducing digital simulation together with hardware-in-loop simulation into teaching practice. To this end, we introduced the Arduino-based exercises described in this paper into classroom activities. Our intention was to better motivate the students and to bring to attention the connection between the lessons being taught and their application in the real world. Many authors share the view that *in contemporary mathematical modeling, the process of translating between the real world situations and mathematics in both directions is one of the essential topics in mathematics education, as research and practice enhance each other through the development of new tools for classroom instruction and across a wide range of disciplines* (Blum and Niss, 1991; Kaufmann and Schmalstieg, 2002; Zbiek and Conner, 2006; Hohenwarter *et al.*, 2008; Blum and Borromeo Ferri, 2009; Niss, 2012).

Subject integration is necessary in goal-oriented problem solving. In our case, this pertains to the use of Arduino-based hardware together with mathematical software (*Mathematica*, GeoGebra) and general-purpose software development environments (Arduino and C# IDEs). A certain knowledge of electrical engineering and physics is also desirable. The role of computers in teaching has been studied for quite some time (Underkoffler; 1969). STEM (science, technology, engineering, and mathematics) education is focused on developing tools and processes for teaching, which integrate concepts that are usually taught as separate subjects in different classes and emphasizes the application of knowledge to real-life situations, enabling teachers to teach mathematics and science much better and more effectively (Burkhardt, 2018; Gonzalez and Kuenzi, 2012; Tomaschko and Hohenwarter, 2017). Providing early exposure to STEM content can ensure that students will continue their interest in STEM subjects through middle and high school up to university level (DeJarnette, 2012; Lavicza *et al.*, 2007; Lavicza *et al.*, 2018; Prensky, 2008).

When it comes to the choice of a hardware platform which is suitable for classroom application, one must consider potential obstacles. Many authors are aware that advanced teaching concepts may fail in practice because of high prices and complexity of hardware and/or software or other similar factors which impact low-income societies differently than highly developed ones. For example: *Particularly in developing countries, teachers are sometimes willing to learn new teaching tools, but, due to various reasons, are not able to implement them in class. Distinctive characteristics of educational systems, culture and financial level also impact the integration of technology in school education* (Bhagat and Chang, 2015; Han *et al.*, 2013; Jezdimirović, 2014; Mainali and Key, 2012). We chose Arduino hardware because it is easy to obtain at a modest price, ubiquitous, well supported and easy to configure for the presented teaching topics.

## 3. Interpolation

### 3.1. *Definition*

We start by introducing the Lagrange polynomial. For a given set of data points $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n),\ \ n = 1,2, \dots$ the Lagrange polynomial is defined as

$$L_n(x) = \sum_{i=0}^{n} y_i \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}.$$

In the simple case of two points ($n = 1$) we get linear interpolation

$$L_1(x) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0} = y_0 + \frac{(x - x_0)(y_1 - y_0)}{x_1 - x_0}.$$

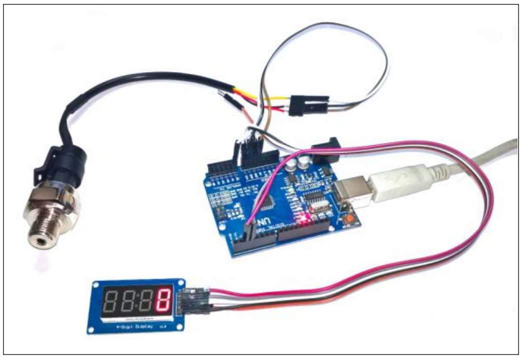


Fig. 1. Arduino with a water pressure sensor and a 7-segment display.

## 3.2. *Real World Example*

Water pressure in a household water supply is essential to the regular function of water heaters and washing machines. To measure water pressure one can use mechanical or electronic devices. An electronic water pressure sensor can simply be installed in a water pipe and connected to Arduino (Fig. 1). The sensor produces output from 0.5 to 4.5V, corresponding to pressures from 0 bar to 12 bar. The dependency is linear.

To measure water pressure, we need to write a C++ program to read the voltage from the sensor and convert it to pressure. The result is displayed on a 7-segment display or otherwise relayed to the user.

## 3.3. *Solution*

First, a formula to convert the voltage from the sensor's output pin to pressure must be constructed. Arduino has analog input pins which can measure voltages from $0V$ to $5V$ and represent them as unsigned 10-bit integers, from 0 to 1023. Knowing the sensor output's range, integer values which correspond to input values of $0.5V$ and $4.5V$ are determined. Interpolating polynomial is constructed for points $(0,0)$, $(5,1023)$, which yields a ratio between the input voltage on the sensor $u$ and analog reading $a = 1023u/5$.

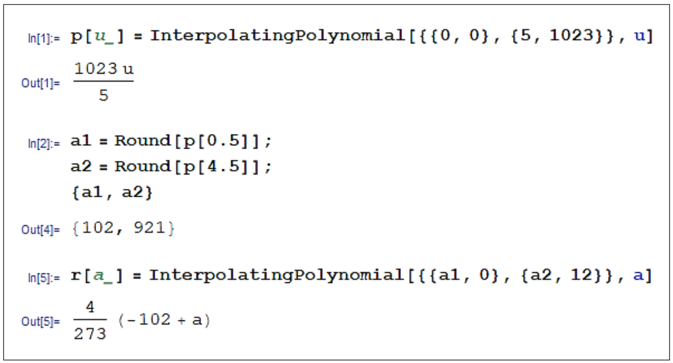
By substituting $u_1 = 0.5$ and $u_2 = 4.5$ and rounding the results, we get $a_1 = 102$ and $a_2 = 921$. These are the readings Arduino will get when sensor voltages are $0.5V$ and $4.5V$ respectively. The readings are converted to pressure by constructing another interpolating polynomial for points $(102,0)$, $(921,12)$:

$$r(a) = \frac{4(a-102)}{273}.$$

The resulting polynomial $r(a)$ converts readings from the analog input pin on the Arduino to pressure in the range from 0 bar to 12 bar. The calculation can be executed in *Mathematica* (Fig. 2).

The result can be obtained geometrically in GeoGebra, by constructing a line through points $A1 = (102,0)$ and $A2 = (921,12)$. The conversion is obtained by placing the point $C$ on the line. By positioning the point $C$ at the $x$ coordinate equal to a value read from the sensor, the corresponding pressure can be read from the $y$ coordinate (Fig. 3). The axes are scaled non-proportionally in this exercise.

It should be noted that a built-in Map function exists in Arduino, which is intended for the same purpose. However, the function is implemented using integer arithmetic and, due to truncation errors, sometimes yields unsatisfactory conversions.



```
In[1]:= p[u_] = InterpolatingPolynomial[{{0, 0}, {5, 1023}}, u]
```

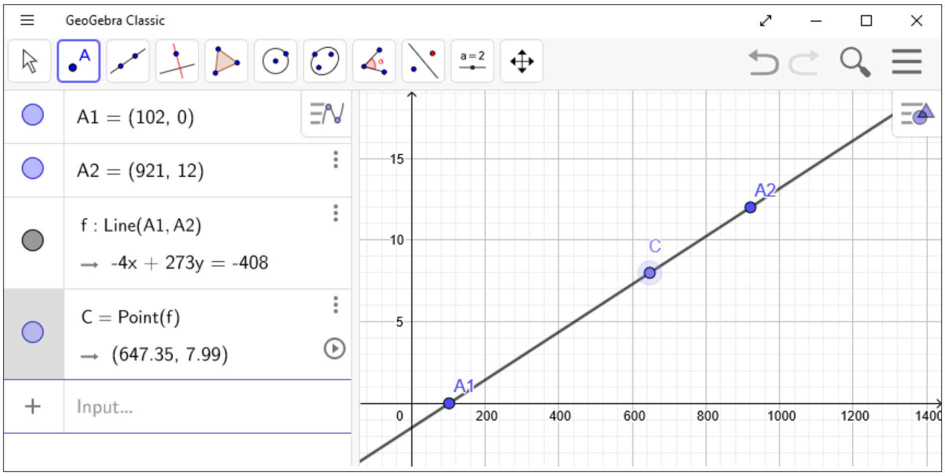
Out[1]= $\frac{1023\,u}{5}$

```
In[2]:= a1 = Round[p[0.5]];
        a2 = Round[p[4.5]];
        {a1, a2}
Out[4]= {102, 921}

In[5]:= r[a_] = InterpolatingPolynomial[{{a1, 0}, {a2, 12}}, a]
```

Out[5]= $\frac{4}{273}(-102 + a)$

Fig. 2. Interpolation in Mathematica to convert sensor readings to pressure.



```
GeoGebra Classic

A1 = (102, 0)
A2 = (921, 12)
f : Line(A1, A2)
→ -4x + 273y = -408
C = Point(f)
→ (647.35, 7.99)
Input...
```

Fig. 3. Conversion of sensor readings to pressure in GeoGebra.

## 4. Numerical Integration

### 4.1. *Definition*

The concept of definite integral is introduced as the problem of area under a graph. The Riemann sum is defined as

$$A = \sum_{i=1}^{n} f(x_i^*)\Delta x_i$$

where $\Delta x_i = x_i - x_{i-1}, \quad x_i^* \in [x_{i-1}, x_i]$. Several special cases are observed, left rule for $x_i^* = x_{i-1}$, right rule for $x_i^* = x_i$ and midpoint rule, for $x_i^* = (x_i + x_{i-1})/2$. In the case of equidistant points, i.e. $h = \Delta x_i$ for $i = 1,2,\dots,n$, the formula becomes

$$A = h\sum_{i=1}^{n} f(x_i^*).$$

If the function $f$ is unknown and only the points $(x_i, y_i)$ are available, $f(x_i^*)$ can be substituted with $(y_i + y_{i-1})/2$ and thus the trapezoidal rule is obtained. Depending on the course level, Newton-Cotes quadrature formulas may also be used.

### 4.2. *Real World Example*

This example presents the working principle of a digital electrical energy meter, similar to those found in households. The example electric circuit consists of a power supply unit and two loads $B1$ and $B2$, which can be independently switched on and off (Fig. 4). The voltage and current are measured by the Arduino, and then the consumed energy is calculated by numerical integration.

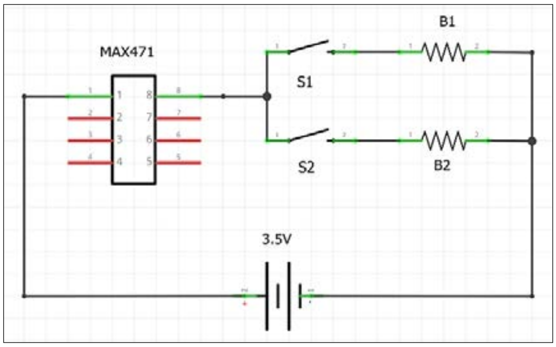


Fig. 4. Schematic for the current measuring experiment

Assuming a constant voltage on the power supply, calculate the energy consumed by the loads over an interval of time, defined by start time $t_1$ and end time $t_2$. During that interval the loads are randomly switched on and off.

For a constant voltage $U$ and a constant current $I$, the power is constant, $P = UI$. For time-dependent values of voltage and current, $u(t)$ and $i(t)$, the power is also time-dependent. Instantaneous power is then calculated as $p(t) = u(t)i(t)$. Both $u(t)$ and $i(t)$ must be defined and available between $t_1$ and $t_2$. In our example, the voltage is constant while the current is not, as it depends on the states of the switches $S1$ and $S2$. The energy consumed by the loads in the circuit is

$$W = \int_{t1}^{t2} p(t)dt = \int_{t1}^{t2} u(t)i(t)dt = U \int_{t1}^{t2} i(t)dt.$$

The average power is obtained as energy divided by $T$,

$$P = \frac{W}{T} = \frac{1}{T}U \int_{t1}^{t2} i(t)dt,$$

where $T = t_2 - t_1$.

## 4.3. *Solution*

Two flashlight bulbs were used as loads and connected to the power supply unit (PSU) through the MAX471 IC (Fig. 5), which measures the current that flows through it and converts it to voltage on its OUT pin, at the ratio of $1V/A$. The voltage at the PSU was set to $3.5V$. By switching the lights on and off during measurement, a series of varying current readings was obtained.

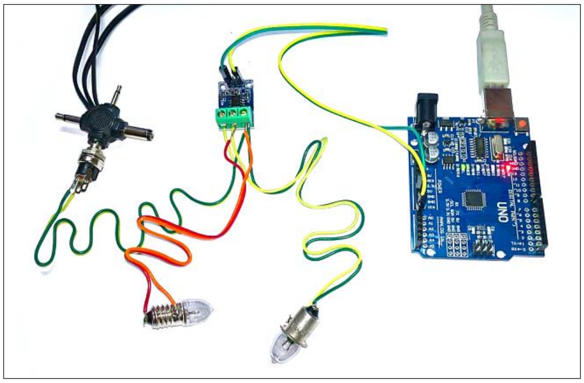


Fig. 5. Setup for the current measuring experiment

The data for $i(t)$ was sampled by the Arduino, the values were relayed to a PC and saved in a file. Then, the power $P$ was calculated by approximating the definite integral by using one of the numerical methods described above.

A simple C++ program was used to read the analog input in 1/4 second intervals and print it to the serial port. Another program was written in C# to collect the data from the serial port on the PC and save it into a file. With the program running, lightbulbs were connected and disconnected several times. Raw data was converted to the range from $0V$ to $5V$ using linear interpolation in the same way as in the previous exercise. The data was scaled and graphed in Mathematica (Fig. 6).

To calculate the average power, the definite integral in

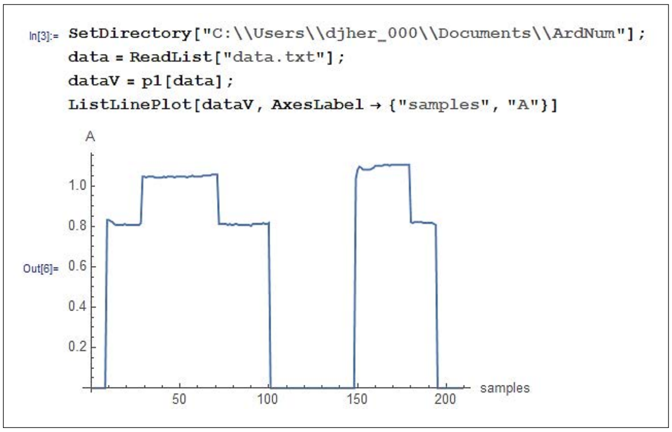
$$P = \frac{1}{T} U \int_{t1}^{t2} i(t)dt$$

was approximated with the trapezoidal rule. Since the raw data was sampled at regular $1/4$s intervals, the equidistant form of the trapezoidal rule was used,

$$A = h \sum_{i=1}^{n} \frac{y_i + y_{i-1}}{2}$$

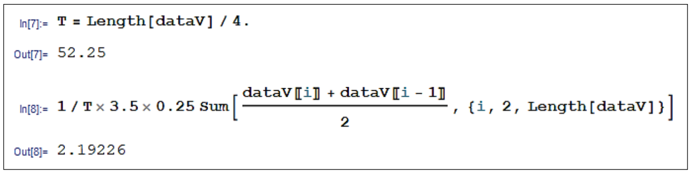
since $h = \Delta x_i = 1/4$ for $i = 1,2, \dots, n$. The power $P$ was obtained as

$$P \approx \frac{1}{T} U \, \frac{1}{4} \sum_{i=1}^{208} \frac{y_i + y_{i-1}}{2} = 2.19226W$$



```
In[3]:= SetDirectory["C:\\Users\\djher_000\\Documents\\ArdNum"];
        data = ReadList["data.txt"];
        dataV = p1[data];
        ListLinePlot[dataV, AxesLabel → {"samples", "A"}]
```

Fig. 6. Converting raw data and graphing the results



```
In[7]:= T = Length[dataV] / 4.
Out[7]= 52.25
In[8]:= 1 / T × 3.5 × 0.25 Sum[(dataV[[i]] + dataV[[i - 1]])/2, {i, 2, Length[dataV]}]
Out[8]= 2.19226
```

Fig. 7. Numerical integration in Mathematica

where $y_i$ is the sample at index $i$, $T = 209s/4 = 52.25$ and $U = 3.5V$. The calculation in Mathematica is shown in Fig. 7.

## 5. Numerical Differentiation and Curve Smoothing

Finite difference formulas for numerical differentiation are approximations of derivatives at a point. They are used to calculate the value of the first derivative of a function for a given $x$, or when the function is unknown and only sampled data is available. Geometrically, a finite difference is the slope of a secant through $(x, f(x))$ and $(x \pm h, f(x \pm h))$, which approximates the value of the first derivative at $x$,

$$f'(x) \approx \frac{f(x \pm h) - f(x)}{h}.$$

A symmetric difference can also be used,

$$f'(x) \approx \frac{f(x + h) - f(x - h)}{2h}.$$

As $h$ gets smaller, the accuracy of the approximation increases, i.e. secant approaches the tangent at $x$.

### 5.1. *Real World Example*

In engineering practice there are many instances where voltage change rates must be measured and calculated, e.g. electric car applications (Kim *et al.*, 2014) or early failure detection in electronic components (Sathik *et al.*, 2018). Since battery charging and discharging may take a long time, it is impractical for a classroom exercise. A capacitor charging experiment may therefore be a good substitute. The aim of this exercise is to illustrate how the voltage change rate can be calculated by means of numerical differentiation. As capacitor charging and discharging are asymptotic processes, we can decide to stop charging when the voltage change rate drops below a certain threshold.

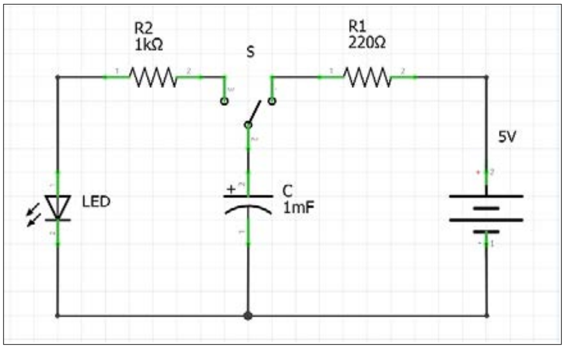


Fig. 8. Schematic for the capacitor charging experiment

Capacitor $C$ is charged through a resistor. After charging, it is discharged through another resistor and a red LED. The circuit is presented in Fig. 8. The user can choose between charging and discharging by toggling the switch $S$. The voltage on $C$ is measured on Arduino. Calculate and graph the charging rate of the capacitor over time. Establish a computable stopping condition for charging, based on the voltage change rate.

During charging, the voltage across the capacitor depends on $R$, $C$ and input voltage $U$ according to the formula

$$U_C(t) = U(1 - e^{-t/RC}).$$

The RC time constant $\tau = RC$ gives time required to charge the capacitor to approximately 63.2% of the charging voltage, or to discharge the capacitor to approximately 36.8% of its initial voltage. As capacitor charging and discharging are asymptotic processes, the time to full charge is infinite. An approximate stopping condition is sometimes established as $t_{\text{stop}} \approx 5\tau$. However, if the capacitance and resistance in the circuit are unknown, one must use a different stopping criterion, based on available numerical data.

## 5.2. *Solution*

If charging voltage $U$, resistance $R$ and capacity $C$ are known, the problem is easily solved in GeoGebra. First, the values $U = 5V$, $R = 1k\Omega$ and $C = 1000\mu F$ are defined and $U_C(t)$ is graphed. The constant $\tau$ is calculated and points $T_1 = (\tau, 0)$ and $T_2 = (5\tau, 0)$ are placed on the $x$ axis. Lines perpendicular to the $x$ axis are placed at $T_1$ and $T_2$ and their intersections with $U_C(t)$ are defined as $A$ and $B$. It is now easily observed that $U_C(\tau) = 3.161$ and the ratio $U_C(\tau)/U = 0.632$ equals 63.2%.

The value $U_C(5\tau) = 4.966V$ corresponds to the voltage across the capacitor at the end of charging. The slope of the graph at that point is found by placing the tangent $S$ to $U_C(t)$ at $t = 5\tau$. In the example in Fig. 10 the slope is 0.033. Different values of $R$ yield different values of the slope. For example, for $220\Omega$ and $R = 1k\Omega$ the slope is 0.153 and 0.003 respectively.

If the values for $R$ and $C$ are unknown or unreliable, a different criterion is devised to stop charging. By sampling the voltage across the capacitor during charging, an array of values $(i, u_i)$, $i = 1,2, \dots$ is obtained. The slope of the tangent to the charging curve at $(i, u_i)$ is observed for $i = 2,3, \dots$ and charging is stopped as soon as it falls below a certain threshold. The slope is calculated numerically by using one of the difference schemes for the first derivative.

The experiment was set up on a breadboard according to the schematic (Fig. 9). The voltage across the capacitor was measured with analog input pins on the Arduino. Power supply on the Arduino was used to charge the capacitor. The capacitor was discharged through a different resistor and a red LED.

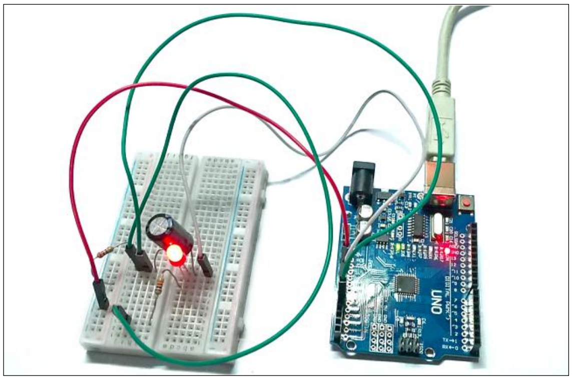
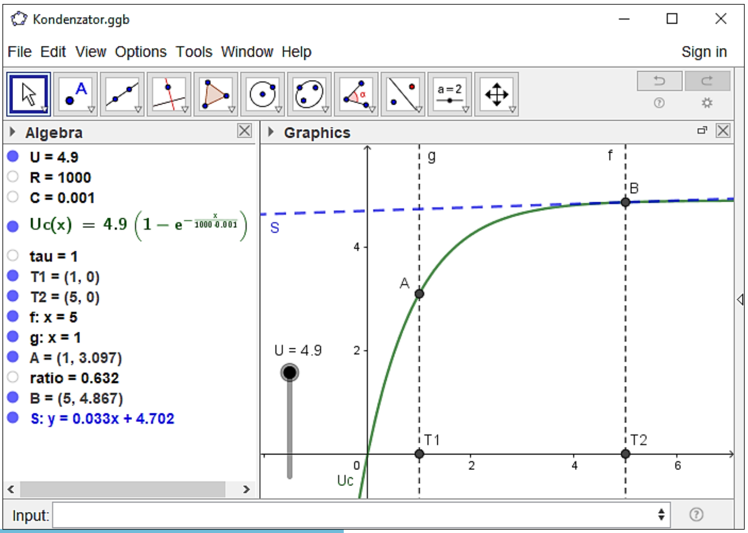


Fig. 9. Setup for the capacitor charging experiment



U = 4.9
R = 1000
C = 0.001
$\mathrm{Uc}(x) = 4.9\left(1 - e^{-\frac{x}{1000 \cdot 0.001}}\right)$
tau = 1
T1 = (1, 0)
T2 = (5, 0)
f: x = 5
g: x = 1
A = (1, 3.097)
ratio = 0.632
B = (5, 4.867)
S: y = 0.033x + 4.702

Fig. 10. Solution of the charging problem in GeoGebra

The experiment was conducted several times, with $220\Omega$, $1000\Omega$ and $10k\Omega$ resistors used for charging. For brevity, only the last case is described. In this case $\tau = 10k\Omega \times 1mF = 10s$. Voltage data was sampled in $1/5$s intervals and saved into a file. The data was read into Mathematica, the values were scaled to the correct voltage range using linear interpolation and graphed (Fig. 11).

The symmetric difference formula was used to calculate numerical approximations to the first derivative at each sampled point except the first and last, and the resulting data was graphed (Fig. 12). Also, sample indices on the $x$ axis were converted to time in seconds.

The graph in Fig. 12 is jagged due to electronic noise, low precision of the A/D conversion and possibly other artifacts. It is obvious that the slope of the tangent decreases with time, however, choosing a simple threshold for the slope would probably not be the best stopping criterion, because a spike due to noise might trigger the criterion too early. A better solution would be to use moving average with appropriate window size or to numerically smooth the sampled curve.

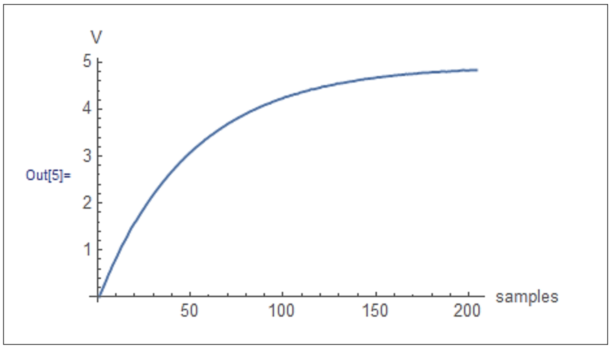
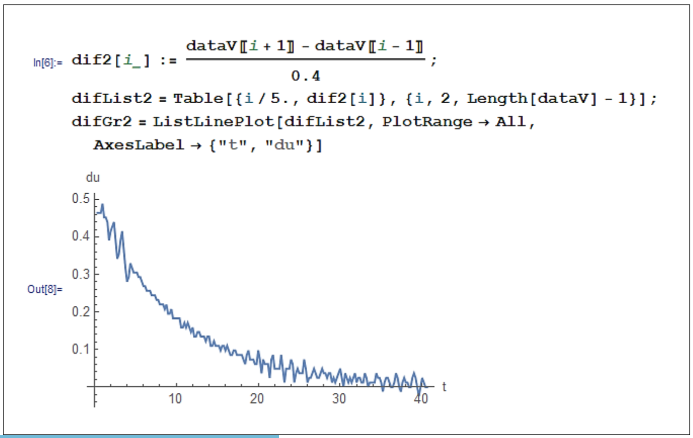
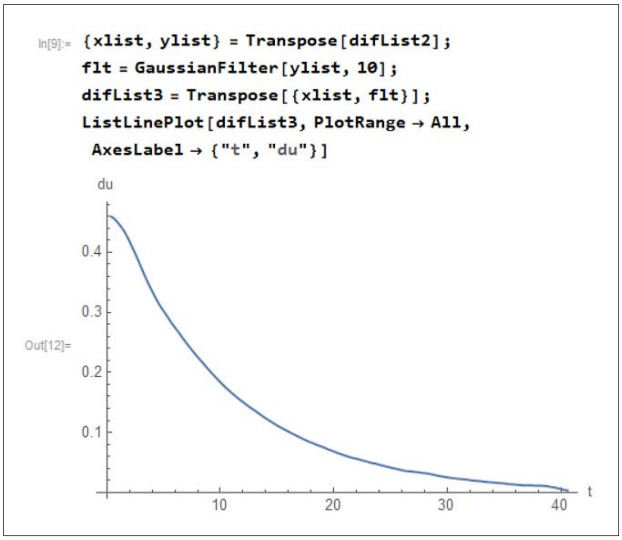


Fig. 11. Voltage across the capacitor while charging.



```
                         dataV⟦i + 1⟧ - dataV⟦i - 1⟧
In[6]:= dif2[i_] := ─────────────────────────────;
                                  0.4
        difList2 = Table[{i / 5., dif2[i]}, {i, 2, Length[dataV] - 1}];
        difGr2 = ListLinePlot[difList2, PlotRange → All,
          AxesLabel → {"t", "du"}]
```

Fig. 12. Result of numerical differentiation.



```
In[9]:= {xlist, ylist} = Transpose[difList2];
        flt = GaussianFilter[ylist, 10];
        difList3 = Transpose[{xlist, flt}];
        ListLinePlot[difList3, PlotRange → All,
         AxesLabel → {"t", "du"}]
Out[12]=
```

Fig. 13. The filtered curve.

### 5.3. *Curve Smoothing*

As discussed in the previous section, the curve obtained by numerical differentiation (Fig. 12) should be smoothed before the stopping criterion is applied. If the analytical form of the curve is known, then an appropriate linear or nonlinear fitting procedure can be applied to fit it to experimental data. If the analytical form is not known, filtering is often applied to remove noise and obtain a smooth curve. We chose to apply the Gaussian filter. The GaussianFilter function already exists in *Mathematica* and is not difficult to program in C#. The curve obtained by filtering is shown in Fig. 13.

With noise removed from the data, the curve is now smooth and the stopping criterion can be applied more reliably.

## 6. Classroom Experiences

The focus of our research was to determine whether the exercises described in this paper help promote student motivation, skills development and knowledge retention. The exercises were introduced into the Numerical mathematics course held at the Faculty of Science, University of Novi Sad, during the school years of 2018/2019 and 2019/2020. We conducted an experiment, described in detail in Section 7, which confirmed our expectations.

The exercises were also used as motivational examples in our other courses: a C# programming course and an Arduino programming course held for students at the Faculty of Science, and also in three courses on Arduino programing, which we held for elementary school and high school teachers during 2018. In these cases, no formal testing was conducted. Instead, we held informal conversations with the students throughout the courses, and the teachers were given a questionnaire, in order to collect their opinions and insights. The feedback from all participants was generally positive, with the majority agreeing that they liked the practical and tangible approach to the subject matter, as well as horizontal curriculum integration. We present some of the most interesting remarks and comments, which will help us improve our approach to teaching in the future.

“**It is frustrating when a program sometimes writes floating-point numbers with a decimal point and sometimes with a comma.**” Upon being explained why this happens, namely, that the decimal separator depends on regional settings of the operating system, the students concluded that “a scientist needs not know such intricate details outside their domain of interest”. In their view, having to write extra code to solve something that should not be a problem in the first place is “extraneous”. This is one example where the flexibility of the software platform, i.e. support for various languages and cultures, effectively becomes an obstacle to the programmer.

“**Integer division is causing numerical errors**” While mapping the values read from Arduino to voltages, most of the students encountered numerical errors stemming from integer division. The students previously took a C# programming course which covered integer and floating-point data types, as well as the differences between integer and floating-point division. Despite that, they were expecting the division to work in the same way as in *Mathematica*, unaware of the implications of using only integer values in C# expressions. This error was easily remedied by casting integers to a floating-point data type. Obviously, theoretical knowledge from previous courses does not always transition to practical knowledge and some repetition is necessary.

“**I don’t like having to dimension the arrays beforehand.**” This statement pertains to the creation of arrays in C#. The programmer can choose between fixed-length arrays and variable-length lists. The arrays provide the fastest element access but must be defined beforehand, while lists can expand as new elements are added, but element access is somewhat slower. We explained the differences between arrays and lists to the students and suggested them to use arrays when execution speed is the primary goal. The lists should be used when the number of elements is not known beforehand, or during development, when flexibility is more important. Even after being explained this, some students maintained that gains in computational speed may not be worth the confusion that stems from two different implementations of arrays in C#.

One interesting consequence of using both lists and arrays is that the number of elements in an array is kept in its Length property, while lists use the Count property for the same purpose. This has led to some confusion among the students, who noticed that *Mathematica* “does it right” by providing a simple Length function.

“**Why isn’t there scalar product in C#**” pertains to C# that the students wrote to calculate convolution necessary for the Gaussian filter. The students preferred having vector and matrix operations already implemented in the language or in libraries, instead of having to write the code by themselves. After we explained that there were, in fact, many available software libraries, the students wanted to know why we avoided using them.

This led to a lengthy discussion about the issues which may arise by including third-party code into one's own project: Is the said code reliable? Can it be easily checked and verified for correctness? Does it carry too much unnecessary weight or further dependencies? In the end, we concluded that, for the purpose of this task, it was easier to write our own code.

Finally, we asked the students to ponder the following points and then form their opinion on the usefulness of C# in solving the presented problems:

- Can you write the solution in C# right away?
- Is it easier to solve one problem in *Mathematica*?
- If you had to solve many problems, or problems with large amounts of data, would you then consider solutions in C# to be the most practical?

The consensus among the students was that, for research, everyday work and study, scientific software such as *Mathematica* and *GeoGebra* is more suitable than a general-purpose programming language. If we shift our focus to high-performance computing and processing of large amounts of data, the balance shifts in favor of languages such as C# and C++.

## 7. The Experiment

An experiment was carried out in order to determine whether our examples promote student motivation, skills development and knowledge retention. We observed 33 students in two parallel groups. There were 16 students in the control group (C) and 17 in the experimental group (E). The control group was instructed in a classical way, with two hours of lecture for each topic, followed by two hours of exercises. Theoretical lessons on interpolation, numerical integration and numerical differentiation were presented in lectures, followed by practice sessions in which the students solved computational problems on paper with the help of *Mathematica*. Data for the problems was arbitrarily generated, without connection to the real world. Lectures for the experimental group were organized around the examples presented in this paper in the same time frame as for the control group. Lectures started with a theoretical introduction followed by demonstrations by the lecturer and hands-on exercises. During practice sessions the students were assigned problems similar to those presented in the exercises described in the paper. Exercises were conducted in a computer lab and data for the experiments was acquired from measurements using Arduino hardware. Problems were solved on computers, using *Mathematica*, Arduino and C# IDEs.

### 7.1. *Initial Testing*

To establish a baseline, a test was given to both groups immediately before the experiment. The students solved computational problems in the classical way, earning a maximum of 100 points. During the test, some students did not complete all the assignments and gave up just as they scored enough points for the passing grade. Comparing the results using the t-test, we confirmed that there was no statistically significant difference ($p = 0.93$ with the 95% confidence level) in test scores between the groups (Table 1).

### 7.2. *Final Testing*

The second test was given at the end of the experiment. Both groups did the test on paper with the help of *Mathematica*. During the test it was obvious that the experimental group was more determined to complete their assignments. Comparing the results for the control and experimental groups, the null hypothesis can be rejected with the 95% confidence level, since $p = 0.043 < 0.05$ (Fig. 2). It can be concluded that there is a statistically significant difference in test scores between the groups.

The results of the experiment highlight the better results of the experimental group in mastering the presented learning topics. We noticed that the motivation was higher in the experimental group, which can be attributed to the more engaging and tangible approach to the subject matter, the use of interesting new technologies and dealing with real-world experimental data.

## 8. Conclusions

Interpolation, numerical integration and numerical differentiation are important topics in Numerical Mathematics, which are often taught only theoretically and demonstrated using some arbitrary data. By employing inexpensive Arduino hardware and mathematics software such as GeoGebra and *Mathematica*, a teacher can help students establish a connection between real world problems and mathematics and motivate them with interesting examples in which discovery plays a significant role. Imperfections and inaccura-

Table 1
Baseline test scores

| Group | No. of students | Average points | Std. deviation | $t$ | $p$ |
|---|---|---|---|---|---|
| C | 16 | 61.25 | 23.95 | 0.08 | 0.93 |
| E | 17 | 60.59 | 21.82 | | |

Table 2
Test scores after the experiment

| Group | No. of students | Average points | Std. deviation | $t$ | $p$ |
|---|---|---|---|---|---|
| C | 16 | 68.75 | 22.61 | -2.11 | 0.043 |
| E | 17 | 83.52 | 15.31 | | |

cies which occur in experiments also stress the difference between theory and practice and highlight the need for a pragmatic way of solving problems. The same applies to the choice of the software for solving problems, as both the specialized scientific software and the general programming languages have their advantages and disadvantages. Obviously, one must decide the priorities. Do we aim for mathematical clarity or speed of computation? Are we willing to sacrifice the simplicity of specialized scientific software for the versatility of a general programming language? The answer to these and similar questions is not simple, and it depends on stated aims and goals.

A significant advantage of the Arduino platform is its low cost, ubiquity and availability of software. Having in mind the positive feedback given by the participants of our classes, as well as the results of the experiment, we are planning towards developing more exercises based on Arduino and integrating them into the curriculum.

## Acknowledgments

This paper has been supported by the Ministry of Sciences and Technological Development of the Republic Serbia, under the grant for project TR 32019.

The authors thank the anonymous reviewers for their insightful suggestions, which helped improve the quality of the paper.

## References

Bhagat, K.K., Chang, C. (2015). Incorporating GeoGebra into Geometry Learning A lesson from India, *Eurasia Journal of Mathematics, Science Technology Education* 11.1, 77–86.

Blum, W., Niss, M. (1991). Applied mathematical problem solving, modeling, applications, and links to other subjects – state, trends and issues in mathematics instruction. *Educational Studies in Mathematics*, Kluwer Academic Publishers, Dordrecht 22, 37–68.

Blum, W., Borromeo Ferri, R. (2009). Mathematical Modelling: Can It Be Taught And Learnt? *Journal of Mathematical Modelling and Application,* 1.1, 45–58.

Burkhardt, H. (2018). Towards Research-based Education. Shell Centre for Mathematical Education Publications Ltd. 1–25. https://www.mathshell.com/papers/pdf/hb_2018_research_based_education.pdf.

Cheng, H., Hao, L., Luo, Z., Wang, F. (2016). Establishing the Connection between Control Theory Education and Application: An Arduino Based Rapid Control Prototyping Approach. *International Journal of Learning and Teaching* 2(1),67–72.

DeJarnette, N. (2012). America's children: Providing early exposure to STEM (science, technology, engineering and math) initiatives. *Education* 133.1, pp. 77–84. DOI: 10.1007/s10649-005-9002-4

Drijvers, P. *et al.* (2016). Uses of Technology in Lower Secondary Mathematics Education. In: *Uses of Technology in Lower Secondary Mathematics Education. ICME-13 Topical Surveys*. Springer, Cham, 1–34. DOI: 10.1007/978-3-319-33666-4_1

Gonzalez, H.B., Kuenzi, J.J. (2012). *Science, technology, engineering, and mathematics (STEM) education: A primer,* `https://fas.org/sgp/crs/misc/R42642.pdf`

Han, O.B. *et al.* (2013). Computer Based Courseware in Learning Mathematics: Potentials and Constrains. *Procedia – Social and Behavioral Sciences* 103, 238–244. DOI: 10.1016/j.sbspro.2013.10.331.

Herceg, D., Herceg, Đ. (2008). Numerical mathematics with GeoGebra in high school, *Teaching Mathematics and Computer Science*, 6/2, 363–378.

Herceg, Đ., Herceg, D. (2009). Definite integral and computer, *The Teaching of Mathematics*, 12(1) 33–44.

Herceg, Đ., Herceg, D. (2010). Numerical Integration with GeoGebra in High School, *The International Journal for Technology in Mathematics Education* – 17(4), 205–210.

Hohenwarter, M., Hohenwarter, J., Kreis, Y., Lavicza, Z. (2008). Teaching and learning calculus with free dynamic mathematics Software GeoGebra, *Proceedings of International Conference in Mathematics Education, Monterrey, Mexico*

Jezdimirović, J. (2014). Computer Based Support for Mathematics Education in Serbia. *International Journal of Technology and Inclusive Education* 3.1, 277–285. DOI: 10.20533/ijtie.2047.0533.2014.0036.

Kaufmann, H., Schmalstieg, D. (2002). Mathematics and Geometry Education with Collaborative Augmented Reality. *ACM SIGGRAPH 2002 Conference Abstracts and Applications*. SIGGRAPH '02. San Antonio, Texas: ACM, pp. 37–41. DOI: 10.1145/1242073.1242086.

Kim, E., Shin, K.G., Lee, J. (2014). Real-Time Discharge/Charge Rate Management for Hybrid Energy Storage in Electric Vehicles, *2014 IEEE Real-Time Systems Symposium,* DOI: 10.1109/RTSS.2014.16

Lavicza, Z. (2007). Factors influencing the integration of Computer Algebra Systems into university-level mathematics education. *International Journal for Technology in Mathematics Education* 14(3), 121–129.

Lavicza, Z *et al.* (2018). Educational experience in teaching mathematics online: a case study on the implementation of GeoGebra in an interactive learning environment. *INTED 2017, 11th annual International Technology, Education and Development Conference*, 110–122.

Mainali, B.R., Key, M.B. (2012). Using dynamic geometry software GeoGebra in developing countries: A case study of impressions of mathematics teachers in Nepal. *International Journal for Mathematics Teaching and Learning*, 1–16. `http://www.cimt.org.uk/journal/mainali.pdf`

Niss, M. (2012). Models and Modeling in Mathematics Education. *EMS Newsletter* December 2012, 49–52. `http ://www.euro-math-soc.eu/ems_education/Solid_Findings_Modelling.pdf`

Prensky, M. (2008). The Role of Technology in Teaching and the Classroom, *Educational Technology* Nov–Dec 2008.

Sathik, M.H., Prasanth, S., Sasongko, F., Poub, J. (2018). Online condition monitoring of IGBT modules using voltage change rate identification, *Microelectronics Reliability* 88–90, 486–492, DOI: 10.1016/j.microrel.2018.07.040

Tomaschko, M., Hohenwarter, M. (2017). Integrating mobile and sensory technologies in mathematics education. *Proceedings of the 15th International Conference on Advances in Mobile Computing & Multimedia*, 39–48.

Underkoffler, M.M. (1969). Computer assisted instruction in college general education mathematics. PhD thesis. Iowa State University, DOI: 10.31274/rtd-180813-1201

Zbiek, R.M., Conner, A. (2006). Beyond Motivation: Exploring Mathematical Modeling as a Context for Deepening Students' Understandings of Curricular Mathematics. *Educ Stud Math* 63.1, 89–112. DOI: 10.1007/s10649-005-9002-4

**Đ. Herceg** is a Full Professor at the Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad. His research is focused on Numerical mathematics and the use of computers in teaching.

**D Herceg** is an Assistant Professor at the Faculty of Technical Sciences, University of Novi Sad. Her research is focused on the modeling and measurement of electromagnetic fields, as well as the use of microcontroller boards in teaching.