Simulation and Analysis of Resistive-Capacitive (RC) Circuits

Mirjana Kocaleva Vitanova [1], Zoran Zlatev [1], Elena Karamazova Gelova [1], Jose Alejandro Ramon Rocha [2], Biljana Zlatanovska [1], Aleksandar Krstev [1]

[1] *Faculty of Computer Science, Goce Delcev University, Krste Misirkov, 10A, 2000 Stip, North Macedonia*
[2] *E.T.S. de Ingenierías Informática y de Telecomunicación, Universidad de Granada – Granada, Spain*

Abstract **– This paper investigates the implementation of a resistor-capacitor (RC) circuit simulation using the C++ programming language, focusing on the capacitor charging process in a first-order transient response. The primary aim is to reinforce theoretical understanding through practical computational modelling and visualization. The simulation employs numerical methods to calculate voltage variations across the capacitor over time, allowing dynamic observation of the circuit's behaviour. To enhance comprehension, the results are visualized using GNUplot, generating clear and exportable graphs that illustrate the transient response in a user-friendly format. The findings indicate that the simulated results closely align with theoretical predictions, confirming the accuracy and reliability of the computational model. Furthermore, visualizing the charging process provides an intuitive understanding of RC circuit dynamics, bridging the gap between abstract theory and practical application. This approach demonstrates the educational value of integrating programming and graphical tools in electrical engineering teaching, highlighting how hands-on simulations can strengthen conceptual learning. Overall, the manuscript supports the notion that combining numerical modelling with visual representation enhances both engagement and mastery of fundamental electronic circuit concepts.**

DOI: 10.18421/TEM152-02
https://doi.org/10.18421/TEM152-02

Corresponding author: Aleksandar Krstev,
Faculty of Computer Science, Goce Delcev University, Krste Misirkov, 10A, 2000 Stip, North Macedonia
Email: aleksandar.krstev@ugd.edu.mk

Received: 08 July 2025.
Revised: 13 November 2025.
Accepted: 28 November 2025.
Published: 27 May 2026.

Keywords **– RC circuit, C++ simulation, theoretical and practical integration.**

1. Introduction

Fundamental circuits such as voltage dividers and RC circuits are cornerstones of electronic engineering education. A voltage divider is a simple linear circuit that outputs a fraction of its input voltage using two resistors in series. It is governed by the well-known formula:

$$V_{\text{out}} = \frac{R_2}{R_1 + R_2} V_{\text{in}} \quad (1)$$

Where, V_{in} is the input voltage, and R_1, R_2 are the series resistances. This static relationship is vital for generating reference voltages and signal scaling in analog circuits, and it forms a foundational concept in electrical circuit education [2], [9]. Additionally, high-voltage applications of voltage dividers are explored in power quality monitoring and measurement systems [1], [10]. Simulation of RC circuits, especially when combined with graphical visualization tools like GNUplot, has been shown to enhance understanding of capacitor charging and transient responses, bridging the gap between theory and practical application [12], [13], [14]. Additional studies confirm that simulation and visualization improve conceptual learning and computational reliability [15], [16], [17]. Further research highlights the pedagogical benefits and effects on students' motivation and performance when using simulators and virtual labs [18], [19], [20].

When a capacitor is introduced into the voltage divider typically replacing one of the resistors and forming a series connection with the remaining resistor and a voltage source, the result is an-RC circuit (Figure 1). This transforms the system into one with dynamic, time-dependent behavior. In DC conditions, the capacitor charges gradually, and the voltage across it follows an exponential curve.

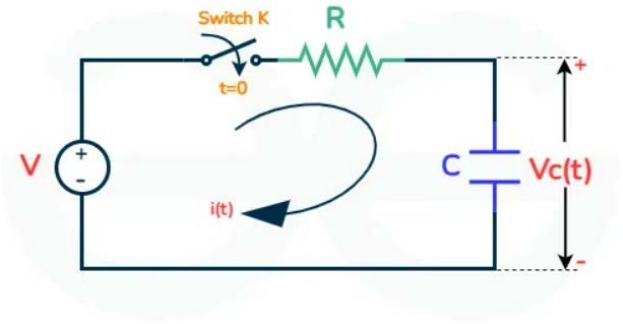

Figure 1. RC circuit [11]

In AC conditions, the RC configuration acts as a low-pass filter, passing low-frequency signals while attenuating higher frequencies [6]. The transient behavior of the capacitor is defined by the time constant, τ=RC, which represents the characteristic time over which the capacitor charges or discharges. After one time constant, the voltage across the charging capacitor reaches approximately 63.2% of the input voltage. Conversely, during discharge, the voltage falls to roughly 36.8% of its initial value, both values stemming from the natural exponential function $e^{-1}\approx 0.368$ [7]. This exponential response distinguishes capacitive circuits from purely resistive ones, in which voltage changes occur instantaneously. The behavior and physical characteristics of capacitors themselves, especially electrolytic types, play a crucial role in determining circuit performance and stability [5].

In frequency analysis, the time constant also determines the cutoff frequency f_c, above which the output of the RC low-pass filter begins to significantly attenuate signals. This cutoff is given by the formula:

$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi RC} \quad (2)$$

Defining the boundary between passed and filtered frequencies. Such filtering behavior is essential in both analog signal processing and digital applications [4].

2. Methodology

The methodology involves developing a theoretical model of the RC circuit, performing numerical simulations to verify its behaviour, and visualizing the results using GNUplot. The implemented code and its logic are explained in detail to ensure transparency and reproducibility of the analysis.

2.1. Theoretical Model of the RC Circuit

The circuit under study consists of a resistor R in series with a capacitor C, connected to a constant DC voltage source V_{in}.

The output as the capacitor voltage $V_C(t)$ was measured. At time $t = 0$, the capacitor is initially uncharged (for a charging scenario), so $V_C(0) = 0$. When the RC circuit is supplied with voltage, an electric current $i(t)$ begins to flow, initiating the charging of the capacitor. The circuit's behavior is governed by two main principles:

- Kirchhoff's Voltage Law (KVL): This law states that the total of all voltage drops in a closed loop must equal the supplied voltage. In this circuit, it gives the equation:

$$V_{\text{in}} = V_R(t) + V_C(t) \quad (3)$$

Here $V_R(t) = i(t)R$ represents the resistor's terminal voltage, and $V_C(t)$ indicates the capacitor's voltage at time *t*.

Capacitor Rule: The current flowing into the capacitor is proportional to the rate at which the voltage across it changes. This is described by: [2], [6].

$$i(t) = C\frac{dV_C(t)}{dt} \quad (4)$$

By substituting this expression for the current into the resistor voltage equation, the KVL equation can be reformulated as:

$$V_{\text{in}} = R\,C\frac{dV_C(t)}{dt} + V_C(t) \quad (5)$$

This equation is a classic first-order linear differential equation, and it models how the capacitor voltage changes over time as the capacitor charges in response to the applied voltage. This differential equation has an analytical solution. Assuming the capacitor is initially uncharged, meaning $V_C(0) = 0$, and the input voltage V_{in} is applied at $t > 0$, the voltage across the capacitor over time is given by:

$$V_C(t) = V_{\text{in}}\left(1 - e^{-t/(RC)}\right), t > 0 \quad (6)$$

This formula indicates that as $t \to \infty$, $V_C(t)$ approaches V_{in} (the capacitor charges up to the supply voltage), and the rate of charging is governed by the time constant $\tau = RC$. At $t = \tau$, $V_C(\tau) = V_{\text{in}}(1 - e^{-1}) \approx 0.632V_{\text{in}}$ (approximately 63.2% of V_{in}). At $t = 5\tau$, the capacitor is over 99% charged (since $1 - e^{-5} \approx 0.993$), which in practice is considered essentially full charge [5], [7]. For completeness, in the discharging case—where a charged capacitor with an initial voltage V_0discharges through a resistor with no external source, the corresponding equation is:

$$V_C(t) = V_0 e^{-t/(RC)} \quad (7),$$

meaning the capacitor voltage decays exponentially from V_0 towards 0 with the same time constant.

After time τ, the voltage falls to ~36.8% of V_0. In the paper, the focus is on the charging scenario, as it closely mirrors the behavior of a step response in a first-order system (and is analogous to the step response of a low-pass RC filter where $V_C(t)$ is the output). The analytical solution expressed by the exponential function is used as the reference for verifying the simulation results [4], [6].

2.2. Numerical Simulation Approach

To simulate the RC circuit in C++, the chosen approach is numerical integration, with the explicit Euler method employed to advance in time and update the capacitor voltage. The Euler method is one of the simplest ways to solve ordinary differential equations (ODEs) approximately: it uses the derivative (slope) at the current point to extrapolate linearly to the next point. Although simple, for sufficiently small-time steps it can produce an adequate approximation of the exponential charging curve [2], [6].

The differential equation is derived from the earlier equation $RC\frac{dV_C}{dt} = V_{\text{in}} - V_C(t)$.

$$\frac{dV_C}{dt} = \frac{1}{RC}\left(V_{\text{in}} - V_C(t)\right) \quad (8)$$

This form clearly shows that the rate of change of V_C is proportional to how far the capacitor's current voltage is from the supply voltage. When V_C is very small (initially 0), the derivative $\frac{dV_C}{dt}$ is large, meaning the voltage will rise quickly. As V_C gets closer to V_{in}, the difference $V_{\text{in}} - V_C$ becomes small, and thus the rate of change slows down, capturing the exponential approach to the final value [7]. Using the Euler integration formula for a derivative $\frac{dV_C}{dt} = f(t, V_C)$:

$$V_C(t + \Delta t) \approx V_C(t) + \frac{1}{RC}\left(V_{\text{in}} - V_C(t)\right)\Delta t \quad (9),$$

where Δt is a small simulation time-step.

The update equation is iterated in a C++ program to simulate the capacitor charging process over time. The choice of Δt must be small enough to ensure stability and accuracy of the simulation. The rule of thumb is to make Δt at most an order of magnitude smaller than the smallest time constant of the system. In this case, with a single time constant τ, choosing Δt as approximately τ/50 or τ/100 effectively captures the circuit dynamics [4], [10]. Using this method allows students to see how a continuous system described by differential equations can be effectively modeled in discrete time using computational techniques bridging the gap between analytical theory and practical simulation [4], [9].

2.3. Data Visualization with GNUplot

Instead of simply displaying numerical values in the console, which can often be difficult to interpret, a graphical representation of voltage versus time is created to enhance clarity and understanding. Visualizing the data through graphs offers a clearer understanding of the exponential behavior during the charging process and makes it easier to compare the results with theoretical expectations by observing the shape of the curve.

GNUplot is employed as the plotting solution. GNUplot is a command-line plotting program that can be driven by external commands or scripts. It is chosen for its simplicity and because it can be invoked from C++ easily, either by calling an external process or by piping commands to it. Specifically, the proposed approach will be [3], [8]:

1. Have the C++ program generates the data for $V_C(t)$ at discrete time steps and save this data to a file (e.g., a CSV or a two-column text file with time and voltage).
2. Automatically call GNUplot from within the C++ program (using a system call or open interface) to plot the data from the file and optionally save the plot as an image (PNG).
3. The graph will then be available for inclusion in reports or for on-screen display.

GNUplot is piping capability that allows us to send plotting commands programmatically. For example, a pipe can be opened to the GNUplot command, and commands such as set title ... or plot "data.txt" with lines can be sent to visualize the data directly from the program. Alternatively, one can prepare a GNUplot script file and invoke GNUplot <scriptfile> via system (). For simplicity, GNUplot is invoked through a one-liner system() command, which avoids the complexity of inter-process communication. It should be noted that using system () to call GNUplot assumes that GNUplot is installed on the system where the C++ code runs.

2.4. Explanation of Code Logic

What the program does is:

1. Set up parameters (R, C, Vin, etc.) and open a file.
2. Loop from t=0 to t=5s, calculate Vc at each step via Euler's method and write the values to file.
3. Close the file and then call GNUplot to create a graph from that file (Figure 2).

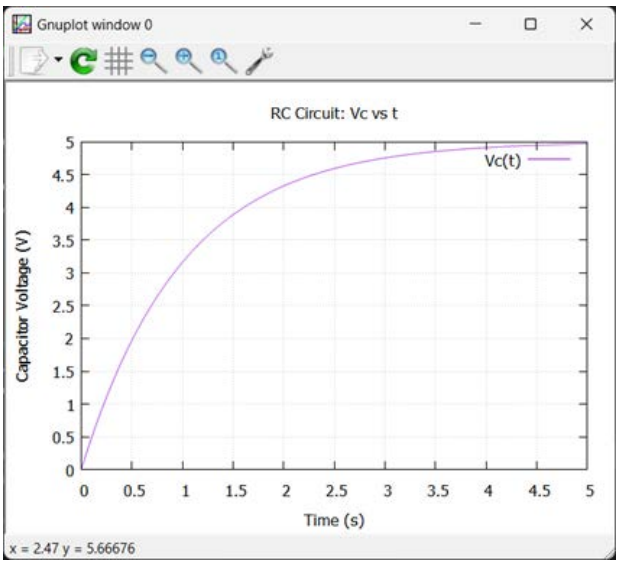

Figure 2. The output graph

It is a straightforward simulation of the RC transient. The decision to use a fixed time step loop (as opposed to calculating at specific analytical points) demonstrates how one can simulate a physical process incrementally. The output file approach decouples the simulation from the plotting, making debugging easier (one can inspect the data file independently) and allows re-plotting with different tools if needed.

To ensure the code is self-contained and functional, it was tested with the chosen parameters. The data file output and the resulting plot were verified to ensure they match the expected theoretical values, as shown in the Results section.

3. Results

The results (Figure 1) confirm the expected behavior: The RC circuit is transient and is well-captured by the simulation. To quantify the accuracy, several data points from the output file may be compared with the exact analytical values. Table 1 presents a comparison of the simulated vs theoretical capacitor voltage at select time instant.

In Table 1, a comparison is presented between the simulated and theoretical capacitor voltage for a 5 V step input applied to a series RC circuit with resistance $R = 1\,\text{k}\Omega$ and capacitance $C = 1\,\text{mF}$. The theoretical values are calculated by $5\left(1 - e^{-t/1}\right)$. The simulation uses Euler's method with dt=0.001 s. The error remains in the order of millivolts, demonstrating the simulation is accurate.

Table 1. Comparison of simulated and theoretical capacitor

Time t (s)	Simulated $V_C(t)$ (V)	Theoretical $V_C(t)$ (V)	Error (Sim – Theoretical) (V)
0.0	0.0000	0.0000	0.0000
0.5	1.9749	1.9673	+0.0076
1.0	3.1698	3.1606	+0.0092
2.0	4.3301	4.3233	+0.0068
3.0	4.7548	4.7511	+0.0037
5.0	4.9671	4.9663	+0.0008

As seen in Table 1, the difference between the simulation and the exact formula is extremely small (on the order of 0.001–0.01 V, or 0.02% of the full scale at worst around 1 s). This indicates that the chosen time step of 1 ms was adequate. If a larger step were used (for example, 0.01 s), the error would increase. Another result to note is the output data file itself. It contains a time series that could be reused for further analysis. For instance, a student could import it into a spreadsheet or MATLAB and do additional plots or calculations (like finding the time where V_C hits 4 V, etc.), thus the program not only visualizes but also provides raw data for other exercises.

Finally, regarding the execution of the program: When run, it produced the console output:

Time constant (tau) = 1 second.

Simulation is completed. Data written to rc_data.txt followed by the opening of a GNUplot graph window with the plotted curve (Figure 1). This interactive window allowed zooming and inspection. If the program is run in an environment without a display (for example, via SSH without X forwarding or on a headless server), the -p flag might not show anything; in those cases, one could modify the gnuplot command to output directly to a file (e.g., a PNG). For example:

"gnuplot -e \"set term png; set output 'rc_plot.png'; plot 'rc_data.txt' ...\"

This would save the plot as *rc_plot.png* without needing a GUI. In this case, since a typical desktop execution is assumed, the interactive graph was the desired result.

The results clearly show that the simulation is working correctly and producing outputs consistent with theory.

4. Discussion

The strong correlation between simulation and theory validates the approach and enhances the understanding of RC circuit behavior. The discussion below highlights several important observations:

1. Exponential Behavior and Time Constant: The RC circuit is exponential charging behavior observed (Figure 1 and Table 1) reinforces the concept of the time constant $\tau = RC$. By $t = 5\tau$, the charging is essentially complete (~99%). This "5τ rule" is often taught to indicate when a transient can be considered settled, and the simulation supports it. The plot's grid and markers can be used to visually confirm these specific percentage points.
2. Numerical Method Accuracy: The Euler method was chosen for its simplicity. It is a first-order method (error per step is $O(\Delta t^2)$, and error accumulated is $O(\Delta t)$). The extremely small errors (in millivolts) in Table 1 indicate that $\Delta t = 0.001$ s was sufficiently small for this problem. If performance was a concern, one could possibly increase Δt to 0.002 or 0.005 and still get acceptable accuracy, but at some point (like 0.1 s, which is 10% of τ) Euler's method would start to show noticeable deviation or even instability if too large. Since computational cost was not an issue for this scale (5000 iterations being trivial for a modern CPU), the approach favored a small step size to ensure high fidelity. To illustrate the effect of step size, one could run the simulation with a larger dt and compare. For example, with dt = 0.1 s, the simulation overshoots a bit and might damp to the final value in a slightly jagged manner. More advanced integration methods (like Runge-Kutta 4th order) could achieve high accuracy with larger steps, but they add complexity. Euler's method proved effective here given a fine step, which is instructive: It shows the balance between computational effort and accuracy.
3. Code Structure and C++ Features: The code presented a straightforward linear structure: parameter setup, loop, file output, then plotting. This linear flow is easy to follow. In a more complex project, one might separate the simulation into a function (e.g., simulateRC () returning a vector of points) and the plotting into another function. It is kept in a single piece for clarity. Some C++ specifics:
    - Basic arrays (essentially writing directly to a file) were used instead of storing the results in a container such as std::vector<double>. The data could have been stored and errors computed on the fly; however, writing directly to a file is more memory-efficient and straightforward.
    - File handling with ofstream was implemented reliably, and error checking on file operations (using if (!dataFile.is_open())) was demonstrated. These checks are important in real-world programming to handle situations such as running in a directory without writing permissions.
4. Use of GNUplot: The integration of an external plotting utility is a practical addition. It shows that one can leverage existing tools to extend the capabilities of a program. By generating the plot automatically:
    - The manual steps required from the user (such as opening Excel or Python to plot the data) are reduced.
    - It provides instant visual feedback, which is valuable for learning and debugging. For instance, if the curve had looked incorrect (say it blew up or oscillated), that would hint at a coding or conceptual error.
    - The presence of a graph also transforms the output from a dry list of numbers into something that can be discussed and included in reports (like this one).
5. Educational Insights: From a learning perspective, writing a simulation in C++ for a simple circuit has multiple benefits:
    - It reinforces the student's understanding of the circuit equations and how they translate into algorithmic steps. The correspondence between the mathematical model and the code (as highlighted in the Implementation section) is explicit.
    - It gives practice with C++ itself (data types, loops, file I/O, calling external programs), thus integrating programming skills with engineering concepts.
    - It exposes the student to numerical methods and the idea that not all equations are solved analytically in practice; often engineers rely on computational methods to predict circuit behavior (especially when circuits get more complex than this first-order case).
6. Extensions and Modifications: The work can be easily extended. Some ideas:
    - Simulate the discharge process: Charge the capacitor fully, then at t=0 switch to a discharge through R (Vin = 0 after some point). This would involve a slight modification: perhaps simulate two phases or use a piecewise Vin (5 V from 0 to some time, then 0 V).

- Include multiple output variables: e.g., also track the resistor's current $i(t)$ and voltage $V_R(t)$. Since $i(t) = CdV_C/dt$, one can output that as well and maybe plot it. The resistor voltage is just $V_R = V_{\text{in}} - V_C$, so it is another interesting curve (it starts at 5 V and decays to 0 V as the capacitor charges, essentially the mirror image of the capacitor's voltage for a series circuit).
- Sweep parameter values: Run the simulation for different R or C values and compare how the time constantly changes the speed of response. This can be done either by multiple runs or by coding loops over parameter values and writing multiple columns or files.
- Compare numerical methods: Implement a second approach (maybe a smaller dt, or a mid-point method) to see the difference.
- Real-world considerations: Discuss how a real capacitor might have leakage or an ESR (equivalent series resistance) which slightly modifies the model; or how the input might not be an ideal step but something with a rise time.

7. Limitations: The simulation has the following limitations:
   - It assumes an ideal circuit. In real life, a 1000 μF capacitor with a 1 kΩ resistor might have some leakage current and the supply might have some internal resistance; those factors are neglected.
   - For extremely large number of steps or very long simulation times, writing to a text file might become slow or produce a huge file. In this case, it is acceptable, as 5000 points represent a negligible computational load. But if one were simulating many transient or high-frequency signals, a binary format or on-the-fly analysis might be needed instead of writing everything out.
   - The use of system () ties the program behavior a bit to the operating system (the command string might need slight changes on different OSes, although gnuplot commands are portable). If GNUplot is not available, the plotting operation fails; this is handled by displaying an appropriate message. For a robust application, one might integrate a plotting library into code (but again, that is a trade-off in complexity).
8. Verification with External Sources: To ensure the simulation is correct, key outcomes can be verified against known references or independent calculations:
   - The shape is monotonic increasing and concave down, as expected for charging RC (no overshoot, no oscillation, since it is a first-order linear system).
   - The time constant measured from the graph (where $V_C \approx 3.16$ V, around 1.0 s) matches the computed $R \cdot C$.
   - The simulation assumes correct physics and mathematics; if necessary, a SPICE tool (e.g., a simple RC netlist) could be used to generate a transient response for comparison. It would match, given the equations are the same.
9. Real-World Application: RC circuits are everywhere (filters, timing circuits, sensor de-bouncing, etc.). By completing this exercise, it not only reaffirms how an RC circuit responds to a step input but also gets practical skills to simulate more complex circuits. For example, the next step could be simulating an RLC circuit (which would involve a second-order differential equation; one could extend to the Euler method or use libraries for solving ODEs). The approach of writing data and using GNUplot would scale to that scenario too, albeit with perhaps more columns (for multiple state variables like capacitor voltage and inductor current).

In conclusion of the discussion, the implemented simulation is verified to be correct. It serves as a template for analyzing simple circuits using computational methods, blending programming and engineering concepts effectively.

5. Conclusion

This paper presented a detailed implementation and analysis of a simple RC electrical circuit using C++. The study successfully met its objective of practically demonstrating a fundamental electronic concept through programming. It showcases a pedagogical approach based on learning by doing, using code to bring circuit equations to life. By integrating electronics theory, numerical methods, and computer programming, the study offers a richer and more comprehensive understanding than any single discipline could provide alone.

The work covered the entire pipeline from theory to results:

- The study begins with the theoretical foundation of the circuit, where the governing equations are derived and the expected behavior, characterized by the time constant $\tau = RC$, is reviewed.

An appropriate numerical method, namely Euler's method, is employed to simulate the differential equation and to justify the selection of parameters such as the time step Δt.

The implementation was done in modern C++, emphasizing clarity and educational value. Each section of code was explained, linking software constructs to physical meaning (e.g., the loop updating V_c corresponds to the capacitor charging over time).

Using GNUplot for visualization was a key practical element, demonstrating how to generate and interpret graphs of circuit behavior directly from a program. Automated plotting facilitated prompt and intuitive analysis of the obtained results.

The results matched the analytical solution closely, validating the approach. The characteristic exponential rise of V_c (t) was observed, and the numerical errors were minimal.

Some of the key takeaways from the paper are:

- Understanding RC Transients:

The simulation clearly demonstrated how a capacitor charges in an RC circuit, reinforcing the concepts of the time constant and the exponential approach to steady state.

- C++ Programming in Engineering Applications:

This paper showed how C++ can be used beyond general-purpose programming to solve real engineering problems. It illustrated the use of file I/O, mathematical operations, and system calls in a practical context.

- Numerical Simulation Skills:

The exercise served as an introduction to numerically solving ordinary differential equations (ODEs), emphasizing key considerations such as step size and accuracy—skills essential for analyzing more complex systems in the future.

- Visualization and Data Analysis:

By integrating GNUplot to generate graphs, the paper highlighted the importance of data visualization. Even a simple circuit's behavior becomes clearer when plotted, and learning to use such tools equips students to better analyze and communicate results.

Future work could involve simulating more complex circuits (e.g., second-order systems, non-linear components), solving systems of equations, or exploring other tools and languages such as Python with libraries like Matplotlib, or domain-specific simulators. Nonetheless, this study demonstrates that even a simple C++ implementation can offer deep insights into electronics fundamentals and serve as a valuable supplement to theoretical learning.

Acknowledgements

The Authors express their gratitude for the support by Goce Delcev University Stip, North Macedonia through an individual scientific fund.

References:

[1]. Anokhin, Y. L., et al. (2017). Application of high voltage dividers for power quality indices measurement. *Electrical engineering and electromechanics*, (6), 53-59. Doi: 10.20998/2074-272x.2017.6.08

[2]. Bird, J. (2017). *Electrical circuit theory and technology*. Routledge. Doi: 10.4324/9781315561929

[3]. Deb, D. (2022). *Gnuplot Helper – a new Utility for gnuplot Graph Plotting Software*. Research Square. Doi: 10.21203/rs.3.rs-548261/v2

[4]. Gochev, V. P., Gocheva, P. V., & Hinov, N. L. (2021). NET implementation of electronic circuit design. *AIP Conference Proceedings*, *2333*(1), 070016. Doi: 10.1063/5.0041606

[5]. Morley, A. R., & Campbell, D. S. (1973). Electrolytic capacitors: Their fabrication and the interpretation of their operational behaviour. *Radio and Electronic Engineer*, *43*(7), 421-429. Doi: 10.1049/ree.1973.0066

[6]. Moya, A. A. (2017). Connecting time and frequency in the RC circuit. *The Physics Teacher*, *55*(4), 228-230. Doi: 10.1119/1.4978721

[7]. Oldham, K. B. (2004). The RC time "constant" at a disk electrode. *Electrochemistry Communications*, *6*(2), 210-214. Doi: 10.1016/j.elecom.2003.12.002

[8]. Williams, T., et al. (1993). Gnuplot 6.0. *Software and User Manual*, *1998*(2004), 2007-2023.

[9]. Xu, M., Sun, J., & Lee, F. C. (2006). Voltage divider and its application in the two-stage power architecture. *Twenty-First Annual IEEE Applied Power Electronics Conference and Exposition, 2006. APEC'06.* Doi: 10.1109/apec.2006.1620584

[10]. Zamora, J. A., Aguilera, E., & Soto, E. (2023). Characterization of a capacitive voltage divider. *Electric Power Systems Research*, *223*, 109635. Doi: 10.1016/j.epsr.2023.109635

[11]. GeeksforGeeks. (2025). *RC Circuit.* GeeksforGeeks. Retrieved from: https://www.geeksforgeeks.org/rc-circuit/ [accessed: 03 July 2025].

[12]. Melchers, H. A. (2019). *Simulation and visualisation of RC-circuits.* [Bachelor thesis, Eindhoven University of Technology].

[13]. Çoramık, M. (2023). Analysis of RC circuits with the help of LabVIEW and data acquisition card for physics laboratory course. *Balıkesir Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, *25*(2), 564-574. Doi: 10.25092/baunfbed.1253712

[14]. Asnaoui, A. H., Kalis, M., & Boulaalam, M. (2024). The study of simulation of RC, RL and RLC circuits using Matlab in teaching. *International Journal of Engineering and Applied Physics*, *4*(2), 955–960.

[15]. Ahmed, F., et al. (2024). Virtual learning via spreadsheets: Real-time simulation of an RC circuit. *American Journal of Physics*, *92*(5), 328-328.

[16]. Radicella, N. (2024). Laboratory teaching proposal on the characteristics of an RC circuit. *Proceedings of NPSE13 Conference*.

[17]. Zavala, G., & Martinez-Torteya, C. E. (2019). Students' Abilities to Solve RC Circuits with Research-based Educational Strategies. *2019 ASEE Annual Conference & Exposition*.

[18]. Yakkou, H., Chillali, A., & El kadri Elyamani, N. E. (2024). The effect of using simulator "evolution of electrical systems''in electricity lessons on students' motivation and academic performance. Heliyon, 10(15). Doi: 10.1016/j.heliyon.2024.e34770

[19]. Torriente-García, I., et al. (2023). Experimenting with RC and RL series circuits using smartphones as signal generators and oscilloscopes. *Revista Brasileira de Ensino de Física*, *45*, e20230143.

[20]. Ahmed, F., et al. (2024). Virtual learning via spreadsheets: Real-time simulation of an RC circuit. *American Journal of Physics*, *92*(5), 328.