Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025 ASEE
Paper ID #48300

WIP: A Novel real-time circuit simulation tool – JSIM

John Francis Simonis, The Ohio State University at Marion
Dr. Qudsia Tahmina, The Ohio State University at Marion

Dr. Qudsia Tahmina, The Ohio State University at Marion

Dr. Qudsia Tahmina is an Associate Professor of Practice at The Ohio State University at Marion and teaches engineering and engineering technology courses. She is involved in curriculum development, assessment of learning outcomes and ABET accreditation.

A Novel real-time circuit simulation tool – JSIM

Abstract

Understanding the behavior of electrical circuits poses significant challenges for today's experiential learners. Traditional teaching methods that rely on static circuit diagrams and manual calculations often fail to engage students who thrive on hands-on learning, programming, and simulation—an approach increasingly prioritized in modern engineering programs [1]–[3]. To address this gap, we introduce JSIM, a real-time circuit simulation tool designed to lower cognitive barriers in circuit analysis while enhancing practical interfacing with hardware through physical breadboarding and low-level programming. Developed exclusively in C++ with an original codebase and optimized for embedded systems, JSIM requires less than 100 KB of storage and achieves significantly faster simulation speeds compared to traditional simulators such as Falstad and MultiSim [5]- [6]. Our testing reveals that JSIM outperforms these simulators in both binary size and output completeness—requiring as little as 42.9 KB on resource-constrained platforms like the RP2350 microcontroller—while delivering accurate nodal and mesh analysis results, including augmented matrices. This reduced binary size makes JSIM a more accessible option for circuit simulation by dramatically lowering hardware requirements and thereby broadening student access. Moreover, unlike Falstad—which fails to provide critical circuit equations—JSIM offers comprehensive outputs that better align with the educational needs of engineering students [2]–[4]. By seamlessly integrating simulation with physical experimentation, JSIM presents a promising avenue for enhancing students' comprehension of circuit theory in modern engineering education. Future research will rigorously evaluate its effectiveness within Electrical Engineering curricula.

Keywords

Mesh analysis, Nodal analysis, Electrical engineering, Circuit simulation, Circuit analysis

Introduction

JSIM is an emerging real-time circuit simulation tool designed to enhance students' understanding of nodal and mesh analysis techniques for basic resistive circuits. By providing circuit analysis feedback within milliseconds, JSIM enables students to iteratively refine their designs in a developmental environment that aligns with modern pedagogical approaches emphasizing experiential learning, hands-on engagement, and low-level programming [1]–[4]. Developed exclusively in C++ with an original codebase and utilizing only the C++ standard libraries, JSIM is highly efficient and uniquely optimized to run on resource-constrained hardware. Unlike traditional simulators such as Falstad or MultiSim—which require significant storage space (at least 75 MB) and advanced operating systems like Windows 10 [5]–[6]—JSIM requires less than 100 KB, making it well-suited for embedded environments such as the RP2350 microcontroller. These lower-cost embedded environments promote greater accessibility in classroom settings.

In addition to its portability, JSIM's reduced computational overhead results in significantly faster simulations, providing immediate feedback in dynamic scenarios like breadboarding labs

where students frequently modify circuit configurations. By dynamically constructing symbolic equations and augmented matrices for nodal and mesh analysis, JSIM creates an interactive learning environment that builds on fundamental programming and mathematical concepts taught in foundational engineering courses [1]–[4]. This unique integration reinforces students' understanding of circuit theory while enhancing their practical programming skills by requiring them to represent circuits via procedural C++ programming within the JSIM simulation framework.

JSIM also employs dynamic memory allocation to ensure space efficiency, enabling it to handle circuits of varying complexity even on hardware with limited resources. While still a prototype, JSIM demonstrates significant promise for incorporation into sophomore-level circuit courses, where its lightweight design and alignment with educational objectives can simplify teaching and foster a deeper understanding of circuit behavior. By combining practical circuit analysis with hands-on programming experience, JSIM represents a novel tool for enhancing engineering education.

Methods

JSIM maintains a lightweight design with portability across platforms. It is purpose-built for helping students with nodal and mesh analysis for resistive circuits, unlike MultiSim and Falstad. This makes JSIM very difficult to compare to traditional circuit simulators. It accomplishes this by leveraging a compact and modular architecture, while also having a fully original codebase.

JSIM is ultimately designed to provide a streamlined and adaptable approach to circuit simulation, distinguishing itself from pre-existing solutions like MultiSim and Falstad through its simplicity and efficiency. Unlike traditional simulators [6], JSIM consists of only eight source files, making it lightweight and highly capable of integrating into any standard **main.cpp** file. Functioning as an advanced library or framework, JSIM offers modular support for both mesh and nodal analysis, enabling developers to compile only the functionalities they require into the final binary. This design minimizes both binary size and memory usage, making JSIM particularly well-suited for resource-constrained environments.

The eight source files that form JSIM are as follows:

- **circuit.cpp** and **circuit.h**: Handle the overall circuit representation and management.
- **elements.cpp** and **elements.h**: Define the behavior and properties of individual circuit elements.
- **mesh.cpp** and **mesh.h** (module): Implement the mesh analysis functionalities.
- **nodal.cpp** and **nodal.h (**module): Provide the nodal analysis functionalities.

For minimal compilation without evaluation capabilities (e.g., simulating and analyzing circuit behavior), only the circuit and elements source files are required. This modularity allows developers and students to tailor JSIM to specific use cases, such as focusing exclusively on nodal or mesh analysis whilst also reducing runtime memory usage and binary size. An added

benefit of this approach is that JSIM is fully extensible using C++ which integrates fundamental programming curricula with Electrical Engineering curricula.

JSIM is also designed to be platform-agnostic, leveraging only the C++ standard library to ensure portability across a wide range of platforms. On Linux-based systems, for example, this can be achieved by using packages like libstdc++. Other embedded hardware platforms, such as the RP2350 microcontroller, include development kits with detailed instructions for installing necessary packages for C and C++ development. By relying solely on the C++ standard library, JSIM maintains compatibility with different compilers as it ultimately requires no external dependencies.

To assess JSIM's effectiveness in educational and practical applications, a comparison was conducted with Falstad, a widely used open-source circuit simulator. The evaluation focused on key metrics:

- **Binary Size:** Space required for the complete compiled program.
- **Correctness of Results**: Accuracy of solutions for nodal and mesh analyses.
- **Completeness of Outputs**: Coverage and clarity of results provided by each tool.
- **Critical Circuit Constraints**: Ability to detect and address constraints like floating nodes and incorrect connections.

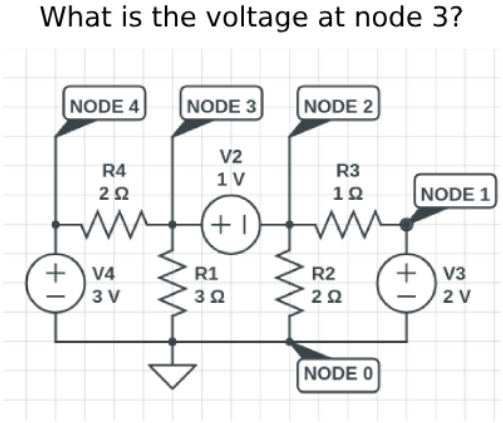

Figure 1: Example of a Supernodal Circuit Problem

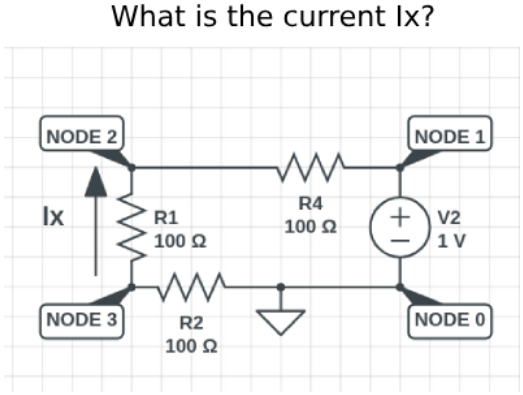

Figure 2: Example of a Simple Resistive Circuit Problem

These two circuit problems were selected to assess the capabilities of each simulator. These circuits were chosen for specific criteria:

- **Figure 1:** This circuit contains a super node and is a suitable candidate for evaluating nodal analysis techniques.
- **Figure 2:** This circuit features only a current source, making it an ideal candidate for testing mesh analysis techniques.

JSIM was compiled for multiple platforms, including MacOS (Intel and ARM), Linux, and Windows—the same platforms supported by Falstad [6]. The binary sizes of JSIM and Falstad were compared across these platforms, with the results expressed as percentage differences. A positive percentage indicates a smaller binary size for JSIM, whereas a negative percentage favors Falstad.

For testing a simple declarative structure for each circuit was used in JSIM where components were placed via their nodal connections. JSIM keeps track of each component through a beginning node, ending node, value, and a label. This allows JSIM to map the position of each component within a netlist as well as retaining its orientation for elements like Current and Voltage sources which change certain values depending on their orientation. An example setup for the circuit shown in Figure 1 can be seen below in Figure 3.

```
#include "circuit.h"
#include "elements.h"
#include <iostream>
#include <vector>
#include <string>

int main()
{
    Circuit test;

    // Circuit configuration
    test.addElement(new Ground(0,"GND"));
    test.addElement(new VoltageSource(0,1,2,"V3"));
    test.addElement(new Resistor(0,2,2,"R2"));
    test.addElement(new Resistor(1,2,1,"R3"));
    test.addElement(new VoltageSource(2,3,1,"V2"));
    test.addElement(new Resistor(0,3,3,"R1"));
    test.addElement(new Resistor(3,4,2,"R4"));
    test.addElement(new VoltageSource(0,4,3,"V4"));

    // Show all circuit connections
    test.displayNetlist();

    // Solve and display nodal
    test.solveNodal();

    return 0;
}
```

Figure 3: Basic JSIM Setup for Supernodal Circuit Problem

Results

To evaluate the effectiveness of JSIM, a comparison was conducted with the Falstad Circuit Simulator using two test circuits: a supernodal circuit (Figure 1) and a simple resistive circuit with a current source (Figure 2). These circuits were selected to assess JSIM's capabilities in performing nodal and mesh analyses, respectively. The evaluation criteria included binary size, correctness of results, and completeness of outputs, as well as the ability to detect critical circuit constraints.

JSIM demonstrated superior performance in several key areas. It successfully produced accurate nodal and mesh analysis results for both circuits, including detailed KCL equations and augmented matrices. By contrast, Falstad failed to generate meaningful equations, limiting its utility as an educational tool for verifying circuit calculations. Additionally, JSIM's ability to produce augmented matrices offers a unique advantage for students learning circuit theory, as it provides a clearer representation of system equations that align with mathematical techniques taught in foundational courses [1] - [4].

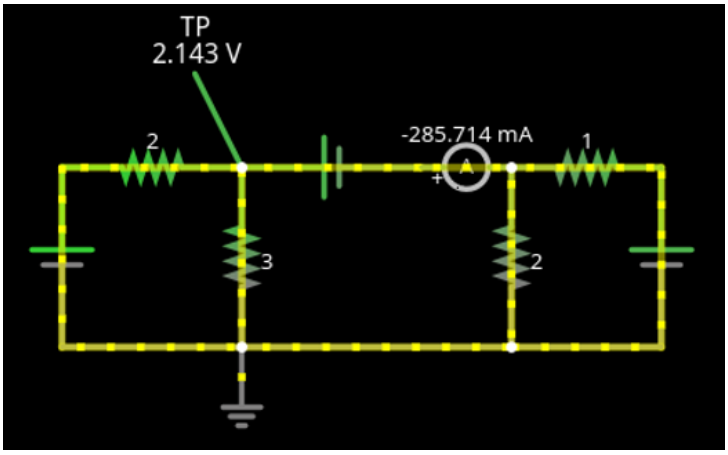

Figure 4: Falstad test for Supernodal Circuit Problem

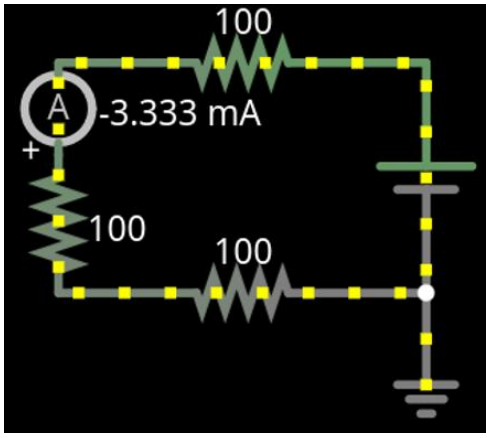

Figure 5: Falstad test for Simple Circuit Problem

```
Netlist:
  Ground GND => nodes(0,-1)  value=0
  VoltageSource V3 => nodes(0,1)  value=2
  Resistor R2 => nodes(0,2)  value=2
  Resistor R3 => nodes(1,2)  value=1
  VoltageSource V2 => nodes(2,3)  value=1
  Resistor R1 => nodes(0,3)  value=3
  Resistor R4 => nodes(3,4)  value=2
  VoltageSource V4 => nodes(0,4)  value=3

========== Symbolic KCL Equations ==========
KCL @ NODE 2 (aka V(2)):
  (V(2) - V(GND))/R2(2Ω) + (V(2) - V3(2V))/R3(1Ω) = 0

KCL @ NODE 3 (aka V(3)):
  (V(3) - V(GND))/R1(3Ω) + (V(3) - V4(3V))/R4(2Ω) = 0

Constraint (Supernode V2): V(3) - V(2) = 1V

========== Augmented Matrix (G|I) ==========
[     1.5000 ,     0.0000 ,    -1.0000 |     2.0000 ]
[     0.0000 ,     0.8333 ,     1.0000 |     1.5000 ]
[    -1.0000 ,     1.0000 ,     0.0000 |     1.0000 ]
=========================================

========== Nodal Solution ==========
V3 = 2.0000 V  (node 1 forced)
V(2) = 1.1429 V
V(3) = 2.1429 V
V4 = 3.0000 V  (node 4 forced)
I(V2) = -0.2857 A
====================================
```

Figure 6: JSIM test for Supernodal Circuit Problem

```
Netlist:
  Ground GND => nodes(0,-1)  value=0
  VoltageSource V2 => nodes(0,1)  value=1
  Resistor R4 => nodes(1,2)  value=100
  Resistor R1 => nodes(2,3)  value=100
  Resistor R2 => nodes(3,0)  value=100

=========== MESH ANALYSIS ===========
Symbolic Mesh Equations:
KVL @ Loop0:
    I1 * R4(100Ω) + I1 * R1(100Ω) + I1 * R2(100Ω) = -V2(1V)

[M|RHS] =
[   300.0000 |    -1.0000 ]

Mesh Currents:
I(Loop0) = -0.0033 A
===================================
```

Figure 7: JSIM test for Simple Circuit Problem

Binary Size Comparison

JSIM's lightweight design is a major differentiator from Falstad. As shown in Table 1, JSIM's compiled binary size is significantly smaller across all tested platforms, requiring as little as 42.9 KB on the RP2350 microcontroller and only 65.9 KB on Linux. In comparison, Falstad requires

between 75 MB and 109 MB, depending on the platform, due to its dependency on the Java Runtime Environment (JRE) [5], [6]. This compact size makes JSIM the only viable option for circuit simulation on resource-constrained hardware like the RP2350.

PLATFORM\PROGRAM	JSIM	Falstad
RP2350	42.9 KB	N/A
Linux (Fedora)	65.9 KB	82 MB
Windows	73.2 KB	75 MB
MacOS (Intel)	81.4 KB	88 MB
MacOS (ARM)	80.3 KB	109 MB

Table 1: Compilation Size Comparison

Completeness of Outputs

JSIM produced comprehensive outputs that included nodal and mesh equations, voltage calculations, and augmented matrices for both test circuits. These results align closely with the educational needs of engineering students, offering a more robust and interactive learning experience. For example, in the supernodal circuit problem (Figure 1), JSIM provided clear KCL equations and accurately calculated node voltages, surpassing Falstad's limited output, which only displayed voltages without associated equations or matrices. It also properly recognized a supernode constraint which modifies the algorithm for standard nodal analysis, something that is largely beneficial to students learning the rules of these techniques for the first time.

Educational Usability

The immediate feedback provided by JSIM, along with its ability to handle dynamic circuit changes efficiently, makes it particularly suitable for hands-on learning environments like breadboarding labs. By delivering detailed mathematical representations and solutions, JSIM can enhance students' comprehension of circuit theory and reinforce programming skills, as highlighted by its alignment with educational objectives in introductory engineering courses [1] - [4]. As a tool, JSIM was explicitly designed to aid students in the process of validation when it comes to calculations.

Future Work

While JSIM shows significant promise, it is still a prototype that is in development and needs rigorous classroom testing. There are several areas where it can be enhanced to broaden its functionality and educational impact:

1. **Support for Time-Dependent Components:** Currently, JSIM lacks the ability to simulate time-dependent components such as inductors and capacitors. Adding support for these elements would enable students to explore transient responses and AC circuit analysis, extending its applicability to more advanced circuit topics.
2. **Graphical User Interface (GUI):** A user-friendly GUI or CLI could make JSIM more accessible to students unfamiliar with basic C/C++ programming,
3. **Integration with Breadboarding Hardware:** Developing compatibility with popular breadboarding platforms like Arduino or Raspberry Pi would allow real-time interaction

between simulated and physical circuits, enhancing experiential learning. This could create an integrated hardware environment for students to breadboard around popular Microcontroller development boards to observe calculated and measured values in the tandem.

4. **Expanded Educational Resources:** Creating tutorials, example problems, and documentation tailored to introductory engineering courses would help instructors integrate JSIM seamlessly into their curricula.
5. **Trial Use in Classroom:** To ensure JSIM is viable in the classroom, JSIM will be integrated and tested on real students to evaluate its full educational potential.

References

[1] The Ohio State University, "First-Year Engineering Program," [Online]. Available: https://eed.osu.edu/node/2130/first-year-engineering-program. [Accessed: Jan. 1, 2025].

[2] University of Michigan, "Engineering 101: Introduction to Computers and Programming," [Online]. Available: https://adue.engin.umich.edu/engr101-computers-programming/. [Accessed: Jan. 3, 2025].

[3] Purdue University, "First-Year Engineering Requirements," [Online]. Available: https://engineering.purdue.edu/Honors/current-students/first-year-engineering-requirements. [Accessed: Jan. 2, 2025].

[4] University at Buffalo, "Computer Programming Requirement - UB School of Engineering and Applied Sciences," [Online]. Available: https://engineering.buffalo.edu/home/academics/undergrad/advisement/computer-programming-requirement.html. [Accessed: Jan. 1, 2025].

[5] National Instruments, "NI Multisim and Circuit Design Suite 14.0.1 Readme," [Online]. Available: https://www.ni.com/pdf/manuals/375231b.html. [Accessed: Jan. 1, 2025].

[6] P. Falstad, "Circuit Simulator (Offline Version)," [Online]. Available: https://www.falstad.com/circuit/offline/. [Accessed: Jan. 12, 2025].