Circuit Analysis with Python

Tiburonboy
juan1543cabrillo@sudomail.com
December 07, 2025

Abstract

This paper presents procedure to analyze electric circuits which may contain resistors, capacitors, inductors, Op Amps, dependent and independent sources using the Python programming language. The procedure presented in this paper will use Modified Nodal Analysis to generate the network equations. It is shown that the SymPy and NumPy libraries can be used to generate symbolic network equations from a circuit's netlist and solve those equations with almost no effort. The procedure is efficient and less error prone compared to manual calculations.

Keywords: Linear Circuit Analysis, Modified Nodal Analysis, Symbolic Analysis, Controlled Sources, Element Stamps

1. Introduction

The purpose of this paper is to describe a procedure implemented in the Python programming language that will automatically generate symbolic network equations from a circuit's netlist. The Python programming language and the associated libraries such as SymPy and NumPy make analyzing electric circuits almost effortless and the JupyterLab Notebooks described in this paper can be used as template for analyzing almost any linear electric circuit.

Circuit analysis is a foundational skill necessary for a broad range of professionals whose work involves the design and development of electronic systems. Electrical Engineers and Circuit Designers are the core audience, as they rely on techniques like Kirchhoff's Laws to create efficient circuits, calculate voltage drops and current flows, and simulate how an electric circuit will behave before it's ever built. Fields like Power Systems Engineering, Control Systems Engineering and even Physics require this knowledge to understand energy distribution, optimize system performance and analyze complex physical models where circuits are used as analogues.

The Modified Nodal Analysis (MNA) is an essential extension of standard Nodal Analysis, and its primary benefits revolve around its systematic and comprehensive nature, particularly for computer-aided analysis. The key advantage of MNA is its ability to accommodate many types of circuit elements in a single, unified matrix formulation. Unlike traditional Nodal Analysis, which struggles to directly model voltage sources and components whose current is not easily defined by node voltages (such as dependent voltage sources and inductors), MNA introduces the currents through these elements as additional unknown variables. This standardization makes the process of formulating the circuit equations highly algorithmic and straightforward to automate, forming the backbone of professional circuit simulators like SPICE. Consequently, MNA eliminates the need for manual, ad-hoc techniques like supernodes or source transformations, allowing for the rapid and reliable analysis of large, complex, and linear circuits.

In 1975, a paper titled, *The Modified Nodal Approach to Network Analysis*, was published by [1]. This was the original scholarly paper on the subject. The analysis method they presented allows for the ability to process voltage sources and current-dependent circuit elements in a simple and efficient manner. The paper describes the formulation of the matrices, the use of stamps and a pivot ordering

strategy. The authors compare their algorithm to the tableau method, a circuit analysis technique, which was an analysis technique being described in scholarly papers at the time.

The analysis of electric circuits, whether done by hand or with the help of Python and MNA, fundamentally relies on the concept of ideal components. These are theoretical abstractions that possess perfect behavior (e.g., a resistor has only resistance, an inductor has only inductance) and ignore the inevitable imperfections of real-world devices, such as parasitic capacitance, lead resistance, or temperature dependence.

Furthermore, almost all conventional circuit analysis methods, including Nodal and Mesh analysis, rely on the assumption of a lumped-element model, which assumes the circuit is small enough that electrical effects occur instantaneously throughout the circuit, neglecting the time delay associated with the propagation of electromagnetic fields. This allows us to treat the circuit's current and voltage as functions of time alone, rather than of both position and time (as required for wave propagation or transmission line effects). Finally, the systematic matrix methods discussed (MNA) are designed specifically for linear time-invariant (LTI) circuits, where the relationship between voltage and current is linear (e.g., Ohm's law holds true) and the component values or connections do not change over time.

Symbolic circuit analysis is a formal technique where the behavior or characteristic of a circuit (like voltage gain or impedance) is calculated with the circuit components and frequency (or time) represented by symbols instead of numerical values.

For small circuits, the analytical expression (e.g., a transfer function) explicitly shows how each component affects the circuit's performance metrics (gain, poles, zeros, etc.). This allows a designer to immediately see which elements are dominant and how to modify component values to meet specifications. A symbolic expression remains valid across a range of component values and operating conditions (as long as the underlying model is valid). By using symbolic analysis the network equations in symbolic form can be useful to document a design or to include in a technical paper.

2. Analysis Method

Modified Nodal Analysis (MNA) was used to obtain the network equations. The MNA procedure provides an algorithmic method for generating a system of independent equations for linear circuit analysis. Once the schematic is drawn and the net list is exported, the network equations and solutions for the node voltages are easily obtained as shown in this paper. A JupyterLab notebook [2] was written to perform the analysis and write the content of this paper. The code in the JupyterLab notebook can be used as a template to analyze almost any type of linear circuit.

The schematic for the example circuits below were drawn using LTSpice and the netlist was exported for the analysis. Drawing a circuit with a schematic capture program and exporting the netlist reduced the possibilities for errors as the analysis proceeds. The analysis begins with generating the network equations from the circuit's netlist by calling:

```
report, network_df, df2, A, X, Z = SymMNA.smna(filter_net_list)
```

The function `SymMNA.smna()` is described in [3]. The SymPy `solve` is capable of working through the algebra to solve the system of equations and can even find the roots of the numerator and denominator polynomials in some cases. Analytic expressions are easily generated for the circuit's node voltages.

Numerical values can be substituted for the component values and NumPy can be used to find numerical solutions for the network equations.

3. Analysis Results

Symbolic analysis presented in this paper employs the MNA technique to generate network equations from the circuit's netlist where the element values are represented by symbols. SymPy is used to solve the system of equations in symbolic form to obtain analytic expressions for the circuit's node voltages. The symbolic expressions can in some cases provide a deeper understanding of how each component contributes to the circuit's operation that can complement numerical simulations.

The netlists for the example circuits are generated by LTSpice from the schematic. The nodes in the circuits were labeled in the schematic, otherwise LTSpice will use default labels such as N001, N002 etc. and the `.smna` function requires integer values for the node numbers and these need to be consecutively ordered with no gaps in the numbering.

Generation of the netlist from a schematic capture program is convenient and less error prone than generating the netlist by hand. A visual inspection of the schematic ensures that the circuit to be analyzed is correct and it follows that the netlist is also correct. This is especially true for larger, more complicated schematics. The procedure for analyzing electrical circuits with Python is best explained by considering the example circuits described below.

3.1. Example 1

3.2. Example 2

3.3. Example 3

4. Discussion and Conclusion

This paper serves as a practical procedure for performing symbolic circuit analysis using the Python programming language and its associated scientific libraries, such as SymPy and NumPy. The primary goal is to provide a systematic and computationally efficient procedure for analyzing linear electric circuits, utilizing JupyterLab Notebooks as adaptable templates. MNA is an essential extension of standard nodal analysis that allows all types of circuit elements (including voltage sources, inductors, and dependent sources) to be incorporated into a single, unified matrix formulation.

Key Concepts:

- Symbolic Analysis: This technique calculates a circuit's behavior (e.g., gain, impedance) with components and frequency represented by symbols instead of numerical values. This provides qualitative insight into the circuit's operation, aids in design optimization, and yields exact analytical solutions that numerical simulators (like SPICE) cannot.
- Modified Nodal Analysis (MNA): The procedure leverages MNA to formulate the circuit problem as a system of simultaneous linear equations, $G \cdot V = I$, which is then solved using Linear Algebra techniques (matrix inversion) implemented in Python.
- Advanced Concepts (The s-Domain): The procedure utilaizes Laplace Transformed Circuit Models, which convert time-domain differential equations into simpler algebraic equations in the s-domain

(complex frequency domain). This is crucial for analyzing circuits with energy storage elements (capacitors and inductors) and determining their transient and transfer functions.

References

[1] C. Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Transactions on Circuits and Systems*, vol. 22, no. 6, pp. 504–509, 1975, doi: 10.1109/TCS.1975.1084079.

[2] Tiburonboy, "Circuit Analysis with Python." Accessed: Dec. 07, 2025. [Online]. Available: https://github.com/Tiburonboy/EE_jupyter_notebooks/blob/main/Circuit_analysis_with_python/Circuit_analysis_w_python.ipynb

[3] Tiburonboy, "Symbolic Modified Nodal Analysis using Python." Accessed: Aug. 13, 2025. [Online]. Available: https://tiburonboy.github.io/Symbolic-Modified-Nodal-Analysis-using-Python/