Design and Implementation Report on Electrical Circuit Analysis Program
EE20084 - Structured Programming

Jake Stewart
JS3910

Electrical and Electronic Engineering
University of Bath
United Kingdom
March 7, 2024

Contents

1 Introduction

This report outlines the development process of an electrical circuit analysis program, as specified in the structured programming coursework. It aims to detail the problem analysis, design decisions, and testing strategies that align with the provided marking rubric.

2 Analysis of the Problem

The main objective of the assignment is to create a program that analyzes electrical circuits using two-port ABCD matrix analysis, with functionalities to read an input ".net" file, perform the analysis calculations, and output the results in the specified CSV format.

2.1 Sub-tasks Identification

- Parsing the input file to extract circuit components, terminations, and desired outputs.
- Malformed input file handling.
- Translating those components into a matrix representation for analysis.
- Calculatinng the ABCD matrix for the circuit.
- Applying the ABCD matrix to the input and output terminations to obtain the desired results.
- Logic error handling and exception raising.
- Generating output files with the analysis results.
- Extending the program to include additional features such as exponent prefixes and decibel calculations.

3 Module realisation and design decisions

This section breaks down the program into modules, classes and functions, and explains the design decisions made for each.
The program is divided into four modules, each with a specific role:

- `net_parser.py`: Parses the input file and extracts circuit information.
- `circuit.py`: Contains classes and functions for circuit analysis.
- `csv_writer.py`: Manages the creation and formatting of the output file with analysis results.
- `main.py`: The main driver script that utilizes the above modules.

Input File Parsing

The parsing of the input file is a critical initial step for the circuit analysis program, involving several key tasks to correctly interpret or sanatise the provided data for circuit components, terminations, and desired outputs. The process is outlined as follows:

1. **Reading the File:** Open and read the contents of the '.net' file, paying special attention to lines starting with the hash symbol (#) as comments and thus skipping them during processing.
2. **Identifying Blocks:** Systematically identify the three main blocks within the file: CIRCUIT, TERMS, and OUTPUT, each of which provides essential data for the circuit analysis.
3. **Extracting Component Data:** Within the CIRCUIT block, extract details of each component, including node connections and component values (resistance, inductance, capacitance, or conductance).
4. **Source and Load Termination:** From the TERMS block, determine the source and load characteristics, including type (Thevenin or Norton) and values.
5. **Output Requirements:** In the OUTPUT block, capture the required output variables and formats, ensuring the program knows what results to calculate and how to format them.
6. **Error Handling:** Implement robust error handling to manage potential issues in the input file, such as missing blocks, incorrect formatting, or unsupported component types.
7. **Storing Data:** Efficiently store the extracted information in the Circuit class module, to the respective component, termination, and output sub-classes.

This structured approach ensures that the program can accurately interpret and process the wide range of circuit definitions it may encounter.

*

Circuit Analysis

Circuit Realization and Solving

To realize and solve the electrical circuit defined in the input file, a systematic approach towards constructing and analyzing the circuit using the ABCD matrix method is as follows:

1. **Circuit Representation:** Construct an internal representation of the circuit from the parsed data, an array of component objects, and the source and load terminations. This can then be sorted based on the node connections to facilitate the ABCD matrix construction.
2. **Matrix Construction:** Implement functions to calculate the ABCD matrices for individual circuit elements (resistors, inductors, capacitors, and any series or parallel combinations thereof).
3. **Cascade Handling:** Develop a method to handle the cascading of components in the circuit, allowing for the multiplication reduction down to an individual ABCD matrix to represent the entire circuit's behavior.
4. **Termination Analysis:** Apply the source and load terminations as defined in the TERMS block to the overall circuit matrix to determine input/output impedance.

5. **Solving the Circuit:** Utilize the inversion of the ABCD matrix of the whole circuit to solve for the desired output variables such as input/output impedance, voltage gain, and power gain.
6. **Error and Exception Handling:** Implement comprehensive error checking and exception handling to ensure accurate results even in cases of complex or unconventional circuit designs.

This detailed, step-by-step approach ensures a robust and flexible circuit analysis tool capable of handling a wide variety of circuit configurations and output requirements.