Mini-Project 2 Phase-1 IR Drop Solver

Taizun Jafri
Arizona State University
tsjafri@asu.edu, jafri.taizun.s@gmail.com

Saivirup Akavarapu
Arizona State University
sakavara@asu.edu

***Abstract*—This report describes challenges faced while developing code for Phase-1 of Mini-Project 2(Static IR Drop Solver)**

I. Project Overview

This static IR drop solver is developed in Python and utilizes Modified Nodal Analysis (MNA) for power grid analysis. It calculates the steady-state voltage at each node by solving the linear system GV = J, which is derived from a SPICE-format netlist representing the power delivery network.

II. Challenges

A. Efficiently Creating G Matrix

In this solver, we implement the modified nodal analysis formula **G*V= J** using matrix multiplication. While creation of the **I** vector (1D matrix) was straightforward, the creation of **G** matrix was confusing at first, which took multiple iterations of code to figure out.

The G-matrix is currently implemented as a SciPy 2D sparse matrix. The size of this matrix is the number of unique nodes and the unique voltage sources.

This addition of rows and columns on the G-matrix because of voltage sources gave us incompatible matrix problems which have since then been solved.

III. Performance Optimizations

- Sparse matrices from SciPy are used for more efficient storage of Voltage(V) and Current(J) vectors.
- G-matrix has been converted into CSR format
- Using python Panda Dataframes for ease of accessing elements at runtime.

IV. Code Snippets

This section explains certain important setions of code that are have been optimized.

```
file_content.append({
   'electrical_component':
       electrical_component,
   'node1': node1,
   'node2': node2,
   'value': component_value
})

file_contents =
    pd.DataFrame(file_content)
```

This snippet shows how each line is stored as a Dataframe so that we may access any node and its corresponding values with ease using only the index of a node or even the name of node.

A. Creating G-matrix

```
G = sp.sparse.lil_matrix((N, N))
J = sp.sparse.lil_matrix((N, 1))
```

We create the 2D G-matrix and 1D current(J) vector using sparse format. This significantly improves memory management and reduces load on system memory. This was tested on the largest benchmark (testcase1.sp) which contained 300,000 nodes but solution was still obtained in less than 15 seconds.

V. Scatter Plot for Test Cases

A. Test Case 17

: SPICE file: testcase17.spout
Output file: testcase17.voltage
===== Analysis Results =====
Mean Error: 0.000000 V
Max Error: 0.000000 V

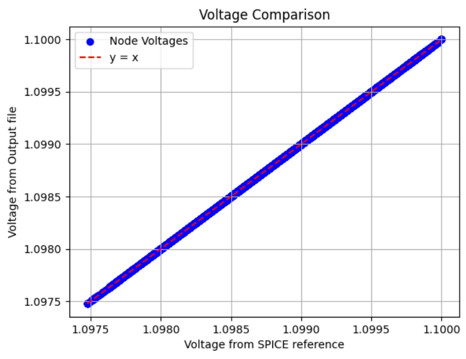

Fig. 1. testcase17 Scatter plot

B. Test Case 18

: SPICE file: testcase18.spout
Output file: testcase18.voltage
===== Analysis Results =====
Mean Error: 0.000000 V
Max Error: 0.000000 V

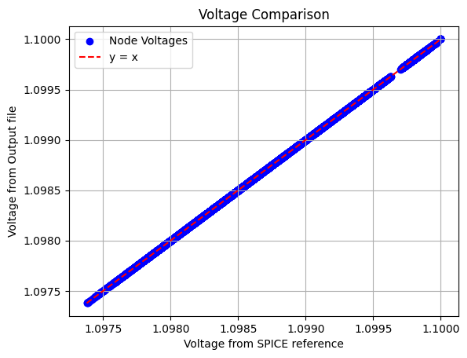

Fig. 2. testcase18 Scatter Plot