AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

CircuitSimulator

z przedmiotu

Języki programowania obiektowego

Elektronika 2021/2022

Michał Nizioł

piątek 13:00

prowadzący: Rafał Frączek

10.06.2022

1. Opis projektu

Symulator obwodów RLC napisany w C++. Rozwiązuje cały układ przy pomocy metody węzłowej. Oblicza spadki napięć i prądy płynące przez każdy element.

2. Project description

Simulator of RLC circuits. Solves circuit with usage of nodal analysis. Calculates current and voltage across all elements.

3. Instrukcja użytkownika

Do obwodu wprowadzamy elementy poprzez podanie typu, numeru końcówek oraz wartości danego elementu. Jeśli dwa elementy mają taki sam numer końcówki tzn. że są ze sobą połączone. Wygodne jest napisanie wcześniej całego układu w pliku tekstowym, a następnie podanie go jako strumień wchodzący do pliku .exe.

4. Kompilacja

Program został napisany w CLion - IDE od czeskiej firmy JetBrains. Kompilacja projektu była wykonywana poprzez narzędzie CMake.

5. Pliki źródłowe

Projekt składa się z następujących plików źródłowych:

- *Circuit.h*, *Circuit.cpp* – deklaracja oraz implementacja obwodu elektronicznego wraz z algorytmami rozwiązywania układu
- *Interface.h, Interface.cpp* – deklaracja oraz implementacja funkcji związanych z interfejsem użytkownika
- *Element.h, Element,cpp* – klasa abstrakcyjna implementująca element z dwoma końcówkami
- *Voltage.h, Voltage.cpp, Current.h, Current.cpp* – klasy implementujące do programu napięcie oraz natężenie prądu
- *Inductor.h, Inductor.cpp* – klasa implementująca cewkę
- *Capacitor.h, Capacitor.cpp* - klasa implementująca kondensator
- *Resistor.h, Resistor.cpp* - klasa implementująca rezystor
- *Source.h, Source.cpp* - klasa implementująca źródło prądowe oraz napięciowe

6. Zależności

W projekcie wykorzystano następujące dodatkowe biblioteki:

- Eigen – biblioteka zawierająca klasy i funkcje powiązane z algebrą oraz analizą matematyczną, w programie wykorzystana do implementacji macierzy zespolonych i dokonywania wydajnych obliczeń związanych z metodą węzłową; https://eigen.tuxfamily.org/index.php?title=Main_Page

7. Opis klas

W projekcie utworzono następujące klasy:

- `Element` – klasa abstrakcyjna zawierająca czysto wirtualne metody oraz pola odpowiedzialne za implementację elementu z dwoma końcówkami
  - `get_node(int node)` – zwraca końcówkę różną od argumentu node
  - `change_node(int node, int value)` – zamienia końcówkę o wartości node na wartość value

- `Inductor, Capacitor, Resistance` – klasy pochodne klasy Element implementujące cewkę, kondensator i rezystor do obwodu, zwierają metody zwracające odpowiednią impedancję, admitancję oraz pola zawierające charakterystyczne właściwości danego elementu (indukcyjność, pojemność, rezystancja)
  - `is_passive` – zwraca true jeśli element jest paswny
  - `get_impedance` – zwraca impedancję elementu
  - `get_admittance` – zwraca admitancję elementu
  - `get_properties` – zwraca wartość charakteryzującą dany element
- `Source` – klasa pochodna klasy Element, implementuje źródła napięciowe i prądowe w obwodzie
  - `get_complex_value` – zwraca wartość źródła w postaci liczby zespolonej po przekształceniu metodą symboliczną
  - `get_type` – zwraca *typ źródła (napięciowe, prądowe, sterowane itd.)*
  - `get_amp`– zwraca amplitudę
  - `get_phase` – zwraca fazę
  - `get_c_freq` – zwraca omegę
  - `get_freq` – zwraca częstotliwość
- `Voltage, Current` – klasy implementujące napięcie oraz natężenie prądu,
  - `get_complex_v` – zwraca wartość źródła w postaci liczby zespolonej po przekształceniu metodą symboliczną
  - `get_amp`– zwraca amplitudę
  - `get_phase` – zwraca fazę
  - `get_c_freq` – zwraca omegę
  - `get_freq` – zwraca częstotliwość
  - `display` – wyświetla ładnie sformatowane dane o tej wielkości

- `Circuit`– klasa reprezentująca obwód elektroniczny, posiadająca metody pozwalające na ustawienie węzłów i gałęzi obwodu, usunięcie elementów poza zamkniętym obwodem oraz rozwiązanie obwodu przy użyciu metody węzłowej (wyjątek stanowi jedno oczko).
  - `set_branches` – przetwarza listę elementów na listy dla poszczególnych gałęzi
  - `get_branch_admittance` – oblicza admitancję gałęzi
  - `get_branch_impedance` – oblicza impedancję gałęzi

- `calculate` – przeprowadza właściwe obliczenia w obwodzie, ustawiając pola odpowiedzialne za przechowywanie informacji o napięciu i natężeniu prądu na właściwą wartość
- `calculate_one_mesh` – przeprowadza obliczenia w wypadku jednego oczka
- `calculate_elements_voltage` – oblicza spadek napięcia na każdym elemencie
- `calculate_elements_current` – oblicza prąd przepływający przez każdy element
- `find_element(int node, int conditio)` – zwraca element o końcówce równej node i różnej od condition

8. Zasoby

brak

9. Dalszy rozwój i ulepszenia

- Dodanie algorytmów rozwiązujących stany nieustalone
- Dodatkowe elementy np. dioda, tranzystory bipolarne, MOSFET
- Graficzny interfejs użytkownika

10. Inne

Symulator ten do rozwiązywania układu używa metody symbolicznej, w której tracona jest informacja o częstotliwości źródła. Dlatego dany obwód, może posiadać źródła tylko z tą samą częstotliwością. Natomiast przez użycie metody węzłowej, nie ma możliwości liczenia układu z samotnym źródłem napięcia.