A.C. CIRCUIT SIMULATION

BRETT YANG

School of Physics and Astronomy

The University of Manchester

Project Report

15 May 2018

ABSTRACT

A command-line program was created in C++ to simulate AC circuits. The main objectives of the program are:

1. Create resistor, capacitor, and inductor as derived class objects from an abstract component class.
2. Store electrical components in a standard template library (STL) container.
3. Build a circuit in series or parallel from existing components stored in the STL container.
4. Calculate and display the magnitude of impedance and phase shift for each component as well as the whole circuit.
5. Display circuit diagram.

1 INTRODUCTION

Alternating current (AC) periodically reverses direction of current flow. In the presence of electrical components such as inductors or capacitors, phase shift between voltage and current arises. The phase shift is caused by an important quantity, impedance, the measure of the resistance to AC circuit. The impedance of a circuit is given by:

$$V = IZ \tag{1}$$

where I is the current, V is the voltage and Z is the impedance.

For an ideal resistor, the impedance takes the value of its resistance:

$$Z_R = R \tag{2}$$

For capacitors and inductors, the impedance takes complex values:

$$Z_C = \frac{-i}{\omega C} \tag{3}$$

$$Z_I = i\omega L \tag{4}$$

where C is the capacitance, L is the inductance and $\omega = 2\pi f$ is the angular frequency of the circuit, which has a dependence on the AC supply frequency, f.

The impedance of a circuit consisting multiple components depends on the way the components are wired together. For a circuit in series:

$$Z_{circuit} = \sum_i Z_i \tag{5}$$

For a parallel circuit:

$$Z_{circuit} = \frac{1}{\sum_i \frac{1}{Z_i}} \tag{6}$$

The phase shift is given by the angle between the real part and imaginary part of the impedance:

$$\varphi = arctan\left(\frac{Im(Z)}{Re(Z)}\right) \tag{7}$$

Note that the arctan function is undefined if $Re(Z) = 0$, in this case, we calculate the phase angle φ under the limit, $\lim_{x\to\pm\infty} \arctan(x) = \pm\frac{\pi}{2}$.

2 CODE DESIGN AND IMPLEMENTATION

2.1 File structure

The consists of two groups: one source code file containing the main program and header files including all class definitions.

The header files include the definition of abstract and derived classes for electrical components (‘component.h’, ‘resisitor.h’, ‘capacitor.h’ and ‘inductor.h’). Information regarding any electrical component is stored in its corresponding class object. Calculations for circuit properties is through the circuit class (‘circuit.h’). The memory block addresses of electrical components are stored in a vector of pointers in a circuit class object. A complex number class enables algebraic operations in the complex domain. The slot class (‘slot.h’) stores a pointer to a component and its location in the circuit. To imitate a real electrical component, a slot stores two vectors of pointers: ‘head’ as wire input and ‘tail’ as wire output. Similar to the components, slot class objects are also stored and accessed from a vector of pointers in circuit class objects.

The of the header files are implemented in the ‘main.cpp’.

2.2 Main program

The program initially sets up the power supply by asking user to key in a peak voltage and a frequency, whereby a circuit class object is created. The program will then print out a menu displaying the available operations.

Should the user choose to create components, the program will ask user to enter the total number of components to be created, then iterates through the steps for creating each component. One requirement is that the user must assign each component a unique name. This allows the identification and retrieval components during circuit assembly. Once components are created, the menu will show an extra option of building the circuit, as shown in Figure 1.

```
.............AC Circuits.............

Please select a power supply:
What is the peak supply voltage (in Volts)?
10
What is the frequency (in Hertz)?
100
Menu:
1. Create component(s).
0. Exit.
1
How many components would you like to add? Please enter an integer:
3
Please choose the type of component you would like to add: Enter 1 for capacitor,
2 for resistor, 3 for inductor
1
Please enter a name for your capacitor:
c
Please enter the capacitance (unit: Farrad):
0.0005
Please choose the type of component you would like to add: Enter 1 for capacitor,
2 for resistor, 3 for inductor
2
Please enter a name for your resistor:
r
Please enter the resistance (unit: Ohm):
10
Please choose the type of component you would like to add: Enter 1 for capacitor,
2 for resistor, 3 for inductor
3
Please enter a name for your inductor:
i
Please enter the inductance (unit: Henry):
0.0035
Menu:
1. Create component(s).
2. Assemble the circuit with existing components.
0. Exit.
|
```

Figure 1. The progression of available options on the menu

The build function allows the user to connect arbitrary number of components in series or parallel. The user will need to enter the number of loops they wish to construct. The function will then iterate through each loop and add a component by asking the user to call a unique componentry name. The user will have the option to add another component or to return to the main menu.

3 VALIDATION AND RESULTS

User inputs are subjected to validations by the program. The validation program will check if the input data type is of the correct type, and if applicable, range checking will be performed. For example, if the user enters a component name that does not exist in the component library, the program will print out an error message as well as a list of existing components name and ask user to re-enter.

```
Loop number 2:
Which component would you like to add in this loop? (enter a name)
10
Error: please enter a component within the current circuit library:
as  asd
as
Would you like to add another component? Enter 0 to exit, or any other key to
continue:
0
```

Figure 2. An example of circuit assembly with input validation.

A run through of the program using a test case of a series RLC circuit is shown in Figure 2.

```
Menu:
1. Create component(s).
2. Assemble/re-assemble the circuit with existing components.
3. Print circuit details.
4. Print circuit diagram.
0. Exit.
3
Circuit:
        Power Supply:
        Peak Volatage: 10 Volt(s)   Peak Current: 0.995194 Amp(s)   Frequency: 100
Hz
            Magnitude of impedance: 10.0483 Phase shift: -0.0980827

        res (r1):
            10 Ohm(s)
            Magnitude of impedance10 Ohm(s) Phase shift: 0 Rad

        cap (c1):
            0.0005 Farad(s)
            Magnitude of impedance3.1831 Ohm(s) Phase shift: -1.5708 Rad

        ind (i1):
            0.0035 Henry(s)
            Magnitude of impedance2.19911 Ohm(s)    Phase shift: 1.5708 Rad

Menu:
1. Create component(s).
2. Assemble/re-assemble the circuit with existing components.
3. Print circuit details.
4. Print circuit diagram.
0. Exit.
4
            ----r1----c1----i1----
            |                    |
            ----------V-----------
Menu:
1. Create component(s).
2. Assemble/re-assemble the circuit with existing components.
3. Print circuit details.
4. Print circuit diagram.
0. Exit.
```

Figure 3. Results from a series RLC circuit

An example of parallel circuit is shown in Figure 4:

```
Circuit:
        Power Supply:
        Peak Volatage: 10 Volt(s)   Peak Current: 533.872 Amp(s)    Frequency:
1000 Hz
            Magnitude of impedance: 0.0187311   Phase shift: -1.57061

        res (r1):
            100 Ohm(s)
            Magnitude of impedance100 Ohm(s)    Phase shift: 0 Rad

        cap (c1):
            0.0035 Farad(s)
            Magnitude of impedance0.0454728 Ohm(s)  Phase shift: -1.5708 Rad

        res (r2):
            100 Ohm(s)
            Magnitude of impedance100 Ohm(s)    Phase shift: 0 Rad

        cap2 (c2):
            0.005 Farad(s)
            Magnitude of impedance0.031831 Ohm(s)   Phase shift: -1.5708 Rad

        ind (i2):
            0.04 Henry(s)
            Magnitude of impedance251.327 Ohm(s)    Phase shift: 1.5708 Rad

        cap (c2):
            0.0035 Farad(s)
            Magnitude of impedance0.0454728 Ohm(s)  Phase shift: -1.5708 Rad

        cap (c3):
            0.0035 Farad(s)
            Magnitude of impedance0.0454728 Ohm(s)  Phase shift: -1.5708 Rad

        ind (i3):
            0.04 Henry(s)
            Magnitude of impedance251.327 Ohm(s)    Phase shift: 1.5708 Rad

        res (r3):
            100 Ohm(s)
            Magnitude of impedance100 Ohm(s)    Phase shift: 0 Rad

        res (r4):
            100 Ohm(s)
            Magnitude of impedance100 Ohm(s)    Phase shift: 0 Rad

        cap2 (c4):
            0.005 Farad(s)
            Magnitude of impedance0.031831 Ohm(s)   Phase shift: -1.5708 Rad

Menu:
1. Create component(s).
2. Assemble/re-assemble the circuit with existing components.
3. Print circuit details.
4. Print circuit diagram.
0. Exit.
4
             --------r4--------c4--------
             |                          |
             -----c3-----i3-----r3-------
             |                          |
             ----r2----c2----i2----c2----
             |                          |
             --------r1--------c1--------
             |                          |
             -------------V--------------
```

Figure 4. Example of a parallel circuit

Finally, deconstructors are called to delete all class objects and pointers stored in STL containers.

4 DISCUSSION AND CONCLUSION

We have demonstrated a command-line program for simple AC circuit calculations through derived class objects corresponding to resistor, capacitor, and inductor. While the program has fulfilled the objectives of this project, there is room for improvements. The program is not capable of performing calculations for unideal components suffering from wire resistance and parasitic effects at high frequencies. While the program can store components in parallel or series, it cannot construct composite circuits with both parallel and serialised components.

In order to build composite circuits, one could construct a function to interpret not only componentry information, but also circuit information from the 'head' and 'tail' vectors. Another way to improve the program is to calculate the voltage and current through each component. This approach relies on reducing the series circuit to three or less components by combining those of the same kind. The possible circuit configurations are then standard cases: R, C, L, RL, RC, LC, RLC. The electric charge differential equations for these standard cases are simple, they are provided in the 'series_connect.h' header file. The voltage and current in between the original components can then be calculated by accessing the original componentry information configuration using the solutions.