May 13, 2017

AC circuit impedence calculator

Object Oriented Programming final project

Dónal Murray
Student ID: 8381250

University of Manchester, Manchester, United Kingdom

Abstract

A program is presented for the manipulation and storage of alternating current circuits. Series and parallel circuits can be constructed using an arbitrary number of resistors, capacitors, inductors or other circuits composed of the listed components. Impedences of all components in the circuit as well as the total impedence of the circuit are calculated and circuits diagrams can be printed. The program allows the user to save the current project to a file and load old projects from a file. The program is designed to fully comply with the current C++ standard and to make use of advanced features of object oriented programming contained within this standard.

Contents

1 Introduction

(brief description of the project you have chosen) [1] Most complicated electronics like computers and smartphones make use of direct current (DC) for their power as the voltage and current are both constant. This means that the components do not have to deal with the stresses and instability associated with an alternating current (AC) power source. AC power sources do not have a voltage which is constant in time; rather it varies periodically between a maximum positive voltage and minimum negative voltage, passing through zero volts along the way. This introduces some difficulties into the circuit theory, when compared with DC. The best way to deal with the mathematics of AC circuits is to use complex numbers. The magnitude of the complex numbers is then taken to find the actual values like resistance and capacitance.

This report aims to outline the design and implementation of a program written in C++ to calculate the impedence of AC circuits consisting of ideal resistors, capacitors and inductors in any combination in series or parallel.

C++ is an object oriented programming language created by Bjarne Stroustrup in 1983

[?]

AC circuit theory

Ohm's law for AC circuits is

$$E = \frac{I}{Z}, \tag{1}$$

where... [?]

2 Code design and implementation

C++ allows for programs to be broken up into multiple files in a bid to reduce the number of lines per file and make the code more readable and more straightforward for the programmer to edit. Separating classes into a separate file allows for modularity; that is the ability to reuse the same classes for different programs. A further feature of this modularity is the ability to separate the interface and implementation of each class. The interface was written in a header file (many conventions exist for the naming of C++ header files but this project exclusively uses the `.h` extension) and consists of the class definition in which the member data are declared and the member functions are prototyped. Header files are included (with the `#include "*.h"` preprocessor directive) but not compiled, whereas implementation files are compiled but not included. The implementation was contained in files with the `.cpp` extension, but like header files, many naming conventions exist. In the design of this project, the functionality was broken into seven header files and seven implementation files which are listed in table 1 along with a short description of their contents.

multiple files
headers

Filename	Description
`main.h` `main.cpp`	defines the `libs` namespace and prototypes all functions for main.cpp
`complex.h`	
`complex.cpp`	
`component.h`	
`component.cpp`	
`capacitor.h`	
`capacitor.cpp`	
`inductor.h`	
`resistor.h`	
`inductor.cpp`	
`resistor.cpp`	
`circuit.h`	
`circuit.cpp`	

Table 1: A list of the files and a brief description of their contents.

classes
vectors
smart pointer
lambda
namespaces
function template
static data

3 Results

(illustrates how code is used including input and output details)

4 Discussion and Conclusions

(include discussion of how code could be improved/extended beyond project)

References

[1] R. Lindner, *LHCb layout_2. LHCb schema_2*, , LHCb Collection.