ICOM 4036 – Programming Languages

Programming Assignment 2

Circuit Analysis with Phasors

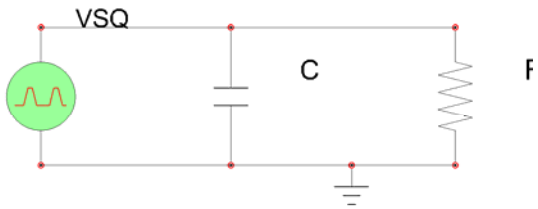
Due on Friday April 2nd and April 9th

Overall Problem Statement

As part of this programming assignment you will implement a library consisting of several subprograms that can be used to analyze electrical circuits containing basic elements including resistors, capacitors, inductors, voltage sources and current sources.

Circuit specification language

Before a circuit can be analyzed it must be encoded as a data structure amenable to computational analysis. For this assignment this data structure will in essence be a matrix of floating point numbers having one row for each circuit element. The columns will represent various characteristics of the element. For instance, consider the following simple electrical circuit of three (3) elements and two nodes.



This circuit should be encoded in matrix form as follows:

Element #	Source Node #	Target Node #	Element Type	Characteristic
1	1	2	1 (voltage source)	10 (V AC)
2	1	2	3 (capacitor)	10E-9 (F)
3	1	2	2 (resistor)	100E3 (Ω)

Node one (1) will always be assumed to be the voltage reference node (a.k.a ground). Nodes are assumed to be numbered in sequence starting at node number 1 and continuing for as many nodes as present in the circuit. Node numbers other than 1 have no particular meaning. Therefore circuit nodes can be numbered in any order.

Circuit elements are also numbered beginning at element 1. Each element is assumed to have two ports, a source port and a target port[1]. By convention the source port is always the one with the smallest number and the one marked with the positive voltage reference (+). Element types are also numeric and can be at least one of the following:

Element Type	Element Type Code	Characteristic Unit
AC voltage source	1	Volts
Resistor	2	Ohm
Capacitor	3	Farads
Inductor	4	Henry

You may add element types to this table if you wish your subroutines to be more general, but this will no be required for full credit.

Your job is to compute the voltage phasor at each node of the circuit. To accomplish this task you should first construct a system of linear equations using nodal analysis. The resulting system should consist of one equation per node and each equation should be expressed in terms of a linear combination of node voltage phasors. To solve the system you may use the code developed in class for solving linear systems of equations. One caveat, remember that since we are dealing with steady state sinusoidal analysis, your currents and voltages can be conveniently represented as complex numbers. You may find Fortran's built-in support for complex numbers quite handy for this exercise.

Your code will consist of several subprograms including the ones shown in the following table:

Subprogram Name	**Description**
`MakLSE`	Constructs the linear system of equations (LSE) from the matrix of components.
`Solve`	Solves the linear system of equations.
`Triang`	Triangulates a matrix representing a linear system of equations on complex variables. Called by `Solve`.
`BkSubs`	Performs back substitution to find the solution of the LSE given a triangulated matrix. Called by `Solve`.

The parameters of the subprograms and their type specifications are part of your design. You may also find it useful to create additional subprograms in order to improve the structure of your code.

Academic Integrity

Note that several details of this project have been intentionally left unspecified in order to provide students with plenty of space for creative design. The end goal is the same for everyone: analyzing electrical circuits using phasors and nodal analysis. There are enough degrees of freedom in order to make it unlikely that two students working independently will end up with

[1] Supporting elements with more than one port is another opportunity for extra creativity.

the same design. All work in tis programming assignment is to be conducted by each student individually. Remember to read the section of the prontuario that deals with academic integrity.

Compiling and Testing Your Code

You will carry out all your software development using the tools available at the Linux Academic Computing Lab (Amadeus). You may work on your personal Linux-based PC, but you must make sure that your code works with the software development tools available at Amadeus as this is the infrastructure that the staff will use to grade your assignment.

Remember from the prontuario that your programming assignments will be graded according to the following late penalty policy:

Days Late	Percent Deduction
1 day late	25%
2 days late	50%
3 days late	100%

Programming assignments will be graded for both correctness and quality according to the following weights:

Criteria	Weight (%)
Correctness	60%
Design	20%
Efficiency	10%
Style & Documentation	10%

Submitting Your Code

You will submit your solution in two installments. The first one is due April 2nd and should include a complete skeleton of the library, although some subroutines may not be completed. The idea is to have a chance to look at you design and make sure that you are not far off schedule. The second installment is due April 9 and should include a fully working library complete with a simple testing program demonstrating that the library works. In both installments you should place all your subroutines into a single Fortran (.f) file and encode it using gzip before submitting it with the submit program that will be available for this programming assignment.

Extra Credit (up to 10 points towards any partial exam)

Implement a plotting subroutine that generates a matrix of characters representing the overlay of the graphs corresponding to an array of phasors. The subroutine should take as arguments the array of phasors to be plotted, the interval in the t (time) axis and the interval in the V/I axis. The subroutine should select a different character symbol for each phasor so as to allow visualization of the different curves on the same plot.