Linköpings Universitet
Institutionen för datavetenskap (IDA)
UPP-gruppen 2015-11-16

Inheritance and polymorphism

Aim

In this laboration you will create an object-oriented framework to build (in C++ source-code) and simulate simple electric networks. You will also use much of what you've learned previously.

Läsanvisningar

- Object orientation
- Object oriented analysis
- Object oriented design
- Classes
  - Inheritance
  - Polymorphism
  - Reference member variables (&)
  - Constant member variables (const)
  - Constant member functions (const)
  - (Static member variables (static))
- Command line arguments
  - C-strings
  - Builtin (C-language) arrays
- Type conversions (to_string, stod, stoi)
  - Catch exceptions from conversion failures (try, catch, exception, std::failure)
- File separation
  - Declaration file
  - Implementation file
  - Include guards

Assignment: Simple Circuit Simulator

Design a set of classes that can work together to build an electric circuit, and a function to simulate the currents and voltages of that circuit.

Components

An electric circuit consists of wires, resistors, capacitors, inductors (coils), diodes, transistors and voltage/current sources. Your simplified implementation is requires to support at least resistors, capacitors and a single battery. It should be possible to connect them in various circuit configurations using C++ source-code.

Circuits

Each component will have two terminals. Component terminals are connected by wires. In our simplified circuits all wire segments that are short-circuit to each other are pulled together to a single point, a connection point. To build a network you should simply create each component and upon creation specify which connection point each terminal is attached to. A list of such connected component will form a circuit.

Simulation

Simulation will run in steps. Each step will advance time by a tiny amount and move some amount of charge through each component in the circuit, from one connection point to the other. The charge moved will depend on the specific behaviour of each component and be proportional to the simulation time step. After a large number of iterations we stop the simulation and examine the final voltages and currents in the circuit.

For simple passive DC circuits (with only a battery, resistors and capacitors) the voltage over each component will stabilize after a large enough number of simulated time units. How many iterations that are required to reach this stable state will depend on the individual circuit and component values. Once the stable state is reached the voltage over each component, and current through each component, should closely match what you can calculate with the normal electric laws.

Simplified component behaviour

In our simplified circuit each component have a somewhat idealized behaviour, and we measure charge in Volts for simplicity. Thus the scientific correctness will be poor. This is our simplified view:

A battery will set the connection point on the positive terminal to the battery voltage, and the connection point on the negative terminal to zero. A battery will never run out.

A resistor will move some charge from its most charged terminal to its least charged terminal. The charge moved will be in proportion to the charge-difference over its terminals, its resistance, and the time step used.

Example: Assume a 2.0 Ohm resistance and simulation in steps of 0.1 time units. Further, terminal a is 5.0 Volt and terminal b at 9 Volt. In this case the potential difference over the resistor is $9.0 - 5.0 = 4.0$ Volt. During the simulated 0.1 time units we move $4.0/2.0 * 0.1$ charges from terminal b (highest potential) to terminal a.

A capacitor will store some charge internally. In each time step it will drain its most positive terminal of some positive charge, and drain its least positive terminal of an equal amount of negative charge. This charge is then stored internally.

The amount of charge moved and stored will depend on the difference in potential, the current charge, the capacitance and the time step used. As you will only simulate simple passive DC circuits where a capacitor is only charging you can ignore the discharge behaviour.

Example: Assume a 0.8 Fahrad capacitance and simulation in steps of 0.1 time units. Further, terminal a is at 5.0 Volt and terminal b at 9.0 Volt. In this case the potential difference over the capacitor is $9.0 - 5.0 = 4.0$ Volt. If it have 3.0 Volt stored we will now store another $0.8 * (4.0 - 3.0) * 0.1$ Volt. Those are pulled from the most positive terminal and stored both internally and on the other terminal.

The voltage over a component is measured by taking the difference between each terminal.

The current through a resistor is obtained by dividing the voltage over it by its resistance. The current through a capacitor is approximated by $C(V - L)$, where C is its capacitance, V the voltage over it and L its charge. Our idealized battery will have zero current.

Hint: You are advised to discuss you final class design with an assistant before you start implementation. This will assure you have a design that are suitable.

Specification of main program

Your main program should accept, in order, the following command line parameters:

- the number of iterations to simulate
- how many iterations (lines) to print
- the time step used in each iteration
- the battery voltage

An example would be to run the simulation in steps of 0.01 time units and 1000000 iterations with a 10 volt battery. Requesting 10 printed lines would mean that one line every 100000 iteration is shown.

Requirement: If any command line argument is missing or is invalid your program should exit with a descriptive error message. Invalid values should be detected by catching exceptions thrown by for example `std::stoi` or thrown by your own functions.

In your main program you should build three circuits according to the figures. You are advised to use one code block for each circuit to avoid mixups. Your class design should make a circuit easy to design. Each circuit should be simulated according to the command line arguments and printed in a table formatted according to given examples. The final output line should match the examples. Minor rounding errors are acceptable, consult your assistant if unsure. If you obtain a difference merely in sign, try to connect the terminals of that component in reverse order.

An example of a code block creating two resistors connected in parallel to a battery and then simulated. ***Note that the object orientation of this example should be improved. Which parts are suitable to move to its own class?***

```
{
  Connection p, n;
  vector<Component*> net;
  net.push_back(new Battery("Bat", 24.0, p, n));
  net.push_back(new Resistor("R1",  6.0, p, n));
  net.push_back(new Resistor("R2",  8.0, p, n));
  simulate(net, 10000, 10, 0.1);
  deallocate_components(net);
}
```

Circuit examples

The top circuit in figure 1 has 4 resistors and a battery. The wire segments (connection points) P, N, R124 and R23 are used to connect the terminals of the components.

The middle circuit have 5 resistors connected in a way where calculation "by hand" is made cumbersome. Unless well versed in such calculations I think it is faster to just write this simulator. As you can see, resistor R1 is attached to P and L, R3 to R and L and so on.

The final circuit also sport some capacitors. For this kind of circuit it would be more interesting if the current or voltage was changing, but our simple simulator do not support that. You can however easily imagine how to improve and extend the simulator to support more advanced circuits and voltage/current sources. As it is we will only look at the final values when the capacitors have stopped charging further.

Voluntary extensions

You may add improvements or features to your simulator at will. Some suggestions:

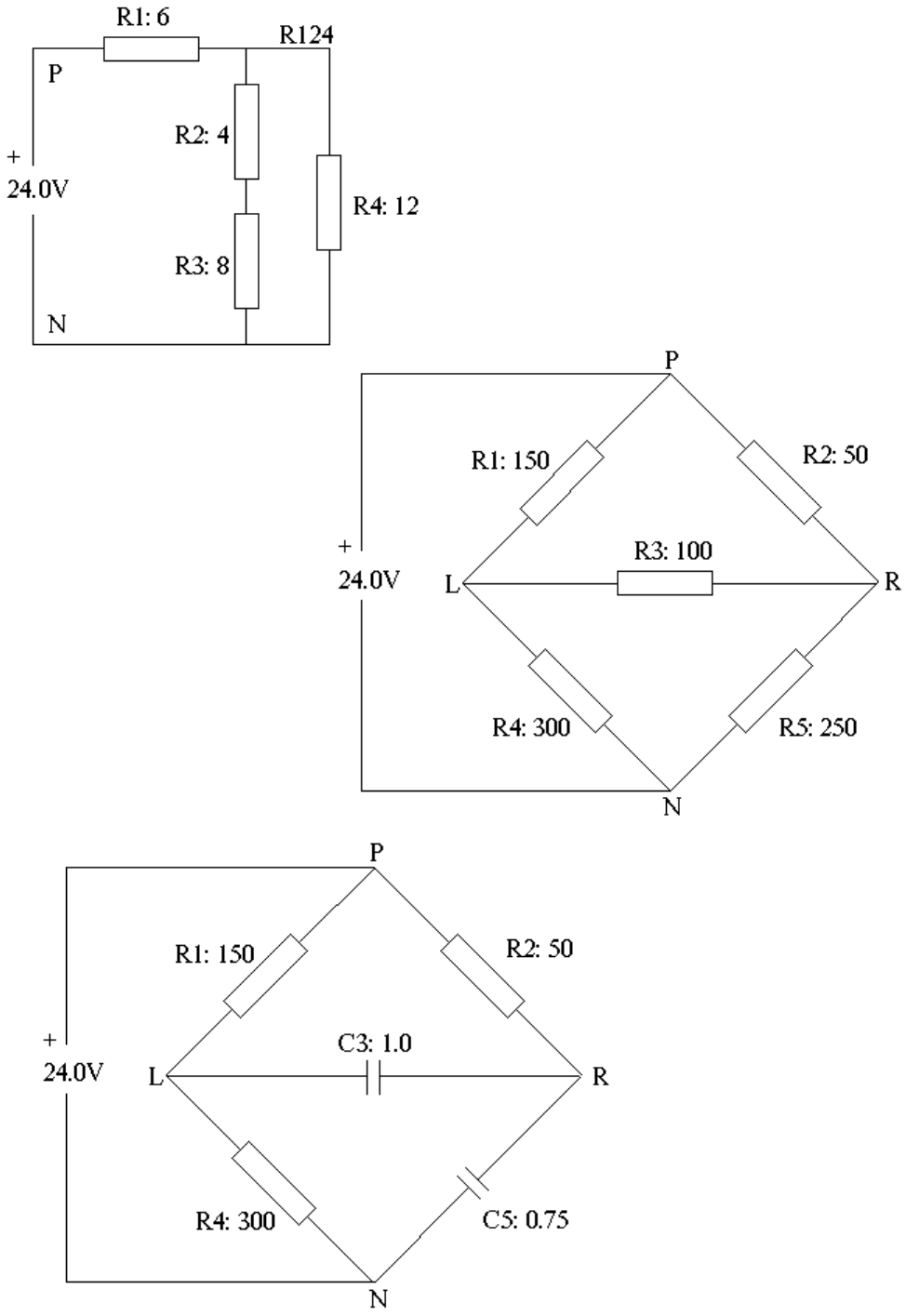
- Ability to load circuits from file instead of adding new circuits as source code and recompile.
- Add more component types.
- Improve the physical model and time accuracy.

```
./a.out  200000 10 0.01 24

          Bat            R1            R2            R3            R4
  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00

          Bat            R1            R2            R3            R4            R5
  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr
 24.00  0.00  7.47  0.05  5.28  0.11  2.19  0.02 16.52  0.06 18.72  0.07
 24.00  0.00  6.40  0.04  4.72  0.09  1.68  0.02 17.60  0.06 19.28  0.08
 24.00  0.00  6.35  0.04  4.69  0.09  1.66  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08

          Bat            R1            R2            C3            R4            C5
  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr
 24.00  0.00  8.50  0.06  4.10  0.08  4.40  0.01 15.50  0.05 19.90  0.03
 24.00  0.00  7.31  0.05  0.94  0.02  6.38  0.01 16.69  0.06 23.06  0.01
 24.00  0.00  7.57  0.05  0.30  0.01  7.27  0.00 16.43  0.05 23.70  0.00
 24.00  0.00  7.79  0.05  0.12  0.00  7.67  0.00 16.21  0.05 23.88  0.00
 24.00  0.00  7.90  0.05  0.05  0.00  7.85  0.00 16.10  0.05 23.95  0.00
 24.00  0.00  7.96  0.05  0.02  0.00  7.93  0.00 16.04  0.05 23.98  0.00
 24.00  0.00  7.98  0.05  0.01  0.00  7.97  0.00 16.02  0.05 23.99  0.00
 24.00  0.00  7.99  0.05  0.00  0.00  7.99  0.00 16.01  0.05 23.99  0.00
 24.00  0.00  8.00  0.05  0.00  0.00  7.99  0.00 16.00  0.05 24.00  0.00
 24.00  0.00  8.00  0.05  0.00  0.00  8.00  0.00 16.00  0.05 24.00  0.00
```



Figur 1: Three example circuits