MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.081—Introduction to EECS I
Spring Semester, 2007

Assignment for Week 7

- Software Lab for March 20th
- Prelab exercises due Thursday March 22th before lab
- Lab on designing and using a virtual oscilloscope.
- Post-lab due Tuesday April 3rd in Lecture.

The constraints view of circuits

In this problem set you will learn about resistor networks both as constraint systems and from a more intuitive perspective. The former is more useful when writing programs to determine the behavior of circuits, the latter is more helpful during design. You will get a chance to use your understanding of resistor networks to design a virtual oscilloscope, and then use that oscilloscope to investigate the behavior of a lego motor.

This problem set has four parts:

1. Post-Lecture lab on using a constraint resolver to solve circuit problems.
2. Tutor problems on analyzing resistor networks.
3. Lab on designing and using a virtual oscilloscope.
4. Post-lab problem on generating constraints for a neuron model.

Note, we are still having some issues with the software for Thursday's lab, so we will post the final version of Thursday's lab Wednesday evening. We will also bring paper copies to Thursday's lab.

Tuesday Post-lecture Software Lab

You will be downloading and using *resolve_constraints.py*, which contains programs for creating lists of constraints and their associated variables, and then once the list of constraints and variables has been generated, determining values for the variables so that the constraints are satisfied. In particular, a user must first create an instance of the class *constraint_list*. Then the user invokes *constraint_list*'s function *add_constraint* multiple times, once for each of the constraints. The function *add_constraint* appends a constraint and the constraint's associated variables to the instance. By calling the instance's function *resolve_constraints*, values are determined for the variables so that the constraints are satisfied. The values of the variables can be printed by calling *constraint_list*'s function *display*.

There are two arguments to the function *add_constraint*. The first is a procedure whose input is a tuple of variable values, and the second is a lists of strings which are the labels for the variables used in the constraint. The order of the strings should match the order of the variables in the tuple. The procedure that is passed to *add_constraint* should return zero when the input tuple satisfies the

constraint, and if the tuple does not satisfy the constraint, the procedure should return a floating point number which indicates how far the tuple is from satisfying the constraint.

As an example, consider the problem of finding values for x and y which satisfy the two constraints

$$5 * x - 2 * y = 3$$

and

$$3 * x + 4 * y = 33.$$

Of course, $x = 3$ and $y = 6$ satisfy the above two constraints.

To use the constraint class and the function *resolve_constraints* to find the constraint-satisfying values of x and y, first define functions which generate the two constraint procedures:

```
def firstEqn():
    # Enforces 5*x - 2*y = 3, assumes [x,y]
    return lambda x : 5.0*x[0] - 2.0*x[1] - 3.0

def secondEqn():
    # Enforces 3*x + 4*y = 33.0, assumes [x,y]
    return lambda x : 3.0*x[0] + 4.0*x[1] - 33.0
```

Second, create an instance of *constraint_list* and add the two constraints, being careful to provide variable labels in the same order as used in the constraint procedures:

```
linSys = constraint_list()
linSys.add_constraint(firstEqn(),['x', 'y'])
linSys.add_constraint(secondEqn(),['x', 'y'])
```

Finally, call *resolve_constraints* and display the solution:

```
solution = resolve_constraints(linSys())
linSys.display(solution)
```

In order to use *resolve_constraints* for circuit problems, one needs an organized approach for generating the variables and constraints for a circuit. In class we discussed the nodal approach for accomplishing this task. The steps in the nodal approach were

1. Label all the circuit node voltages and element currents (noting direction), and select a reference node.

2. For each element, write constitutive equations that relate element currents to the voltages at the element's terminals.

3. For each circuit node, except the reference node, write a conservation law. That is, insist that the sum of currents entering a node should be equal to the sum of currents leaving a node.

In order to use the constraint resolver to solve circuit problems, it is helpful to have functions which return constraint procedures associated with a circuit's constitutive equations and conservation laws. In the file *circuit_constraints.py*, there are functions to generate procedures that implement circuit related constraints. The *resistor* and *vsrc* functions return procedures which implement the constitutive relations associated with a resistor and a voltage source, the *kcl* function returns a procedure which implements the constraint that the signed sum of currents must equal zero, and the *set_ground* returns a procedure which implements a constraint forcing a value to zero (typically used to force the reference node voltage to be zero).

Download and test the constraint System

The software for this prelab is at the 6.081 web site. Download and unzip constraint resolver software, and note that there are four python files: *resolve_constraints.py* and *circuit_constraints.py* described above, as well as two example circuit files, *circuit.py* and *circuit2.py*. Try running the two example circuit files, and notice the second example generates an error. Fix the error in the second example circuit file (the reference node voltage has not been set), then draw the resistor and voltage source circuit diagrams associated with the two examples.

Checkpoint: 4:00 PM

- Show your LA your circuit diagrams, and demonstrate that you have fixed the problem in the circuit file.

Use the constraint resolver

Use the constraint resolver to find the node voltages in the following circuit, assuming $Vs = 5.0$ and $R1 = R2 = R3 = R4$.