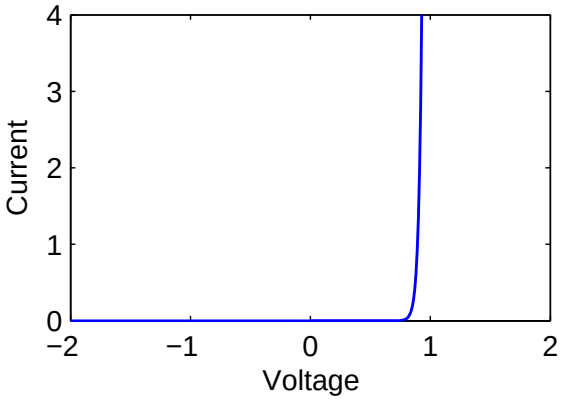
15-884/484 – Problem Set #3

1. **Power Electronics [25pts]** In this question you'll build a circuit to power an LED light in a few different ways. Unlike normal incandescent bulbs, LEDs are trickier because we need to ensure that just the right amount of voltage is applied to the LED. The circuit symbol for an LED looks the same as that for a diode, but with additional lights signifying the emitted light:

In class we said that a diode "permits current to flow only one way," but this is really a simplification. In reality, a diode just gives a relationship between current and voltage, much like a resistor; but whereas a resistor related voltage and current in a linear manner (i.e., via Ohm's law $V = IR$), a diode relates them in a non-linear manner via Shockley's equation

$$I = I_S \left(\exp(V/(nV_T)) - 1\right)$$

Don't worry about any of the terms here except V and I (the rest are just physical constants of the diode, and if you're curious you can look these up). Here's what the relationship looks like for a example diode

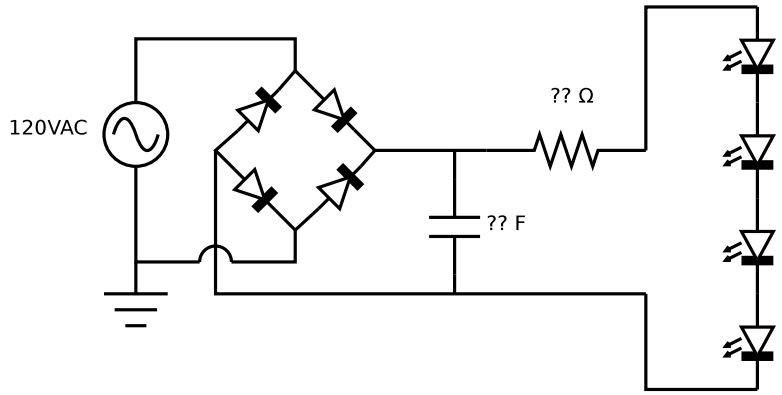


LEDs only emit light for a relatively narrow range of currents: if too little current passes through the LED, then it will not light up, whereas if too much passes through, it will burn out. Thus, for our purposes the take-home message of the diode equation above is that we have to ensure a very precise voltage drop across the LED diode for it to stay lit.

In this problem you will build a rectifier circuit (in QUCS) that converts AC to DC electricity to power a series of LEDs. For all the problems, you can use the elements from the `q1.sch` file, which include 120V RMS AC power source, a transient simulation block (some of the step size parameters can be a pain to set, so we have set up the basic block already), and a diode that models the LED as well as a diode to use for the rectifier (for those who are curious, these models are based on the 1N4003 diode for the rectifier, available here `http://www.digikey.com/product-detail/en/1N4003-T/1N4003DICT-ND/42102` and the TLHK5800 LED, available here `http://www.digikey.com/product-detail/en/TLHK5800/751-1119-ND/1681254`). The goal of this problem is to power a series of 4 of these LEDs using a 120V RMS AC outlet. This LED is meant to function at 20mA (0.02 amperes), where it has a voltage drop of 2V. Thus, for a series 4 of LEDs, we want to maintain a voltage drop of 8V across the series.

For all these questions, include a copy of the QUCS simulator schematic and readings with your solution.

(a) One way to provide 20mA of current through the LEDs is to rectify the 120V AC voltage to 120V DC voltage, then put a resistor in series with the LEDs to ensure that only 20mA pass through. Here is a schematic for this setup:



Starting from the `q1.sch` file, lay out this circuit in QUCS. Find values for the capacity and resistor such that at least 18mA and at most 22mA of current pass through the LED at all times. You can plot the current flowing through the LEDs by adding a current probe and a Cartesian plot.

Hint: If you don't have the right resistance and capacitance, the diodes can easy generate a lot or a very little bit of current. Instead, note that at the desired values (20mA @ 2V, but *just* at these points) the diode behaves like a 100Ω resistor; thus, for getting the right ballpark figure for the the resistance and capacitance, you can replace the LEDs with such resistors, then switch back to the LEDs for the real evaluation.

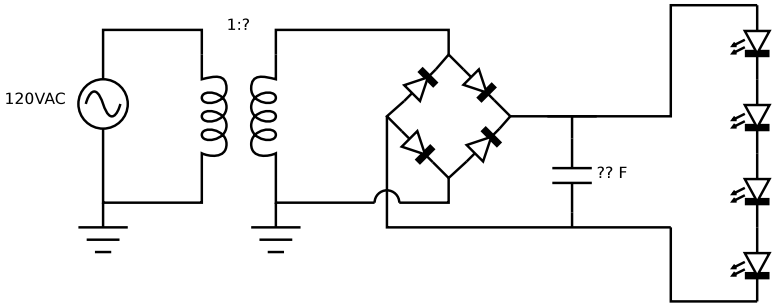
(b) How much power does the above circuit consume? The easiest way to measure this is to put current and voltage probes at the AC source, and take the average of their product over the simulation window. You can then add an "Equation"

in QUCS (the f(u) = u+4 icon in the toolbar), setting power to be the pointwise product of voltage and current and then computing the average of this value, i.e.

```
power = Pr2.Vt*Pr3.It
mean_power = avg(power)
```

(where "Pr2" and "Pr3" would be the names of the voltage and current probes respectively).

(c) Now consider an alternative setup, where we first step down the voltage to its desired level using a transformer, then rectify the signal and run it across the LEDs, without the need for a resistor in series. The circuit looks like this:



Find values for the transformer ratio and capacitor that again ensure at least 18mA and at most 22mA pass through the LED. Use the same hint as provided in part (a); also, be sure to ground both sides of the transformer as shown in the diagram.

(d) Compute the power consumption of this circuit, using the same technique you used in part (b).

(e) Briefly explain why there is or is not a difference in power consumption between the two different circuits.

2. **Power Flow Solver [25 pts]** In this problem you'll implement a simple power flow solver using Newton's method for some standard benchmark problems. In particular, you'll look at (a slightly simplified version of) the IEEE power flow test cases, which have been the standard benchmarks for power flow analysis for some time. To load one of the examples, you can use the command (included with the data)

```
[Y,s,v,slack,pv,pq,mva] = load_cdf(filename);
```

where `filename` is one of the IEEE test cases included in the data (e.g. `ieee14cdf.txt` through `ieee300cdf.txt`) In the resulting variables `Y` is the bus admittance matrix, `s` and `v` are the complex power and voltage (which may be fixed depending on the bus type) and `slack`, `pv`, and `pq` are a list of indices which specify which buses are which types (`mva` is a scaling factor that lets you scale the resulting powers to MW).

Implement a power flow solver in MATLAB using Newton's method, in the following form

```
function [s,v] = pf_newton(Y,s0,v0,slack,pv,pq)
```

where the output s is equal to $s0$ for PQ loads (and the real parts are equal for PV generators), the voltage magnitudes in v are equal to those in $v0$ and the real component of s is equal to $s0$ for PV generators, and the voltage v equals $v0$ for the slack generator.

Report the resulting power and voltage for the 14 bus test case, as well as the number of Newton iterations needed. Also run the 30, 57, 118, and 300 bus test cases, but for brevity just report the average voltage and power in these cases. In all cases include a printout of your MATLAB code.

3. **DC Power Flow Approximation [25 pts]** As discussed in class, DC power flow is a linearized approximation to power flow, where we assume that, line resistances are zero, voltage magnitudes are equal to one, and voltage angles are small, so that $\cos\theta \approx 1.0$ and $\sin\theta \approx \theta$ for all θ. In this problem, you will derive the simplified expressions for DC Power Flow, as well as evaluate it's accuracy compared to the non-linear power flow equations.

   (a) Use the above approximations

   - $R_{ij} = 0 \ \ \forall i, j$
   - $\hat{v}_i = 1 \ \ \forall i$
   - $\cos\theta \approx 1, \ \ \sin\theta \approx \theta \ \ \forall\theta$

   to simplify the power flow equations

$$p_i = \hat{v}_i \sum_{k=1}^{n} \hat{v}_k (G_{ik}\cos(\theta_i - \theta_k) + B_{ik}\sin(\theta_i - \theta_k))$$
$$q_i = \hat{v}_i \sum_{k=1}^{n} \hat{v}_k (G_{ik}\sin(\theta_i - \theta_k) - B_{ik}\cos(\theta_i - \theta_k))$$

   In particular, show that under the DC Power Flow assumptions

$$p = -B\theta, \ \ q = 0$$

   where B is the imaginary part of the admittance matrix formed using these approximations.

   (b) Given a set of real power injections p, how would you determine the corresponding set of phase angles θ? Remember that you have to ensure that $\theta_{\text{slack}} = 0$ (the slack generator), and be sure not to invert any non-invertible matrices. Write your solution as a MATLAB function in the form

```
function theta = pf_dc(B,p,slack)
```

   (c) In this problem you'll compare the AC and DC Power Flow solutions. You'll use the data from the IEEE 14 bus test case, which you can load with

```
[Y,s,v,slack,pv,pq,mva] = load_cdf('ieee14cdf.txt');
```

You'll use both the `pf_newton` and `pf_dc` functions you developed above for this part (if you have errors in these functions, it won't count off for this problem if the remaining code is correct).

i. Determine the B matrix from the Y matrix of the 14 bus example. Hint: Note that this is not just $B = \text{Im}\{Y\}$ since Y contains both resistances and reactances; you'll want to first find these resistances and reactances and set the resistances to zero before forming B.

ii. Evaluate the accuracy of DC power flow in predicting the phase angles. For this part, you can use the following template (this randomly adds addition power consumption to various nodes in the 14 bus network):

```
randn('state',1);
for i=1:1000,
  s0 = s + 0.3*randn(n,1);
  [s1,v1] = pf_newton(Y, s0, v, slack, pv, pq);
  theta = pf_dc(B, real(s0), slack)
end
```

Plot the maximum and mean absolute errors $|\hat{\theta}_i - \theta_i|$ for each bus i, where $\hat{\theta}_i$ is the voltage angle from the DC Power Flow approximation and θ_i is the voltage angle from the AC Power Flow solution.

Note: Although it should not happen for any of the cases above, there are some situations where AC power flow will note converge, and Newton's method will iterate forever; to prevent this, limit the number of Newton steps for your AC power flow solver to 20.

4. **(15-830 only) Optimal power flow [25 pt]** In this problem you'll solve an optimal power flow problem via sequentially calling a solver like YALMIP. In theory, YALMIP can handle general non-linear constraints, so that we could simply input the original optimal power flow problem, and it would find a solution. In practice, however, YALMIP can be quite brittle for non-convex problems (and even for some poorly conditioned convex problems, as you saw in the past assignment), so that it is reasonable to look at other approaches to the problem. Throughout this problem, we'll adopt the notation from the lecture notes, and treat z as a variable that includes all the problem data

$$z = \begin{bmatrix} \theta \\ \hat{v} \\ p \\ q \end{bmatrix}$$

In this problem we'll consider the 14 bus IEEE test case, but slightly modified so that only nodes 1, 2, and 8 are generators (you can load this modified version from the data file `ieee14cdf_q4.txt`).

In the following, we let $SLACK$ denote the slack generator, PV denote the set of PV generators, and PQ denote the set of PQ loads; we'll let GEN denote both the slack

and PV buses, $GEN = PV \cup SLACK$. Assume the following costs for the different generators

$$\begin{aligned} c_1(p_1) &= 5p_1 + 2p_1^2 \\ c_2(p_2) &= 2p_2 + 4p_2^2 \\ c_8(p_8) &= p_8 + 5p_8^2. \end{aligned}$$

and assume that each generator has a maximum capacity of 100MW real power (which equals 1.0 in the units of the 14 bus example) and 38 MVAR reactive power (0.38 in problem units). We'll still assume, as in direct power flow computation, that the voltages for the slack and PV generators are specified as problem input.

Thus, the final optimal power flow problem is given by

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \sum_{i \in GEN} c_i(p_i) \\ \text{subject to} \quad & \hat{v}_i = \hat{v}_i^0, \ \ \forall i \in GEN \\ & \theta_{SLACK} = 0 \\ & p_i = p_i^0, \ \ \forall i \in PQ \\ & q_i = q_i^0, \ \ \forall i \in PQ \\ & p_i \leq 1, \ \ \forall i \in GEN \\ & q_i \leq 0.38, \ \ \forall i \in GEN \\ & s = \text{diag}(v)\bar{Y}\bar{v} \ \ \text{(power flow constraint)} \end{aligned}$$

where $\cdot^0$ denotes the input variables to the typical power flow problem (i.e., the outputs of the `load_cdf` call).

(a) Write a MATLAB routine to solve this problem by iteratively linearizing the power flow constraint, as discussed in the lecture notes, and solving the resulting (now convex) QP using YALMIP.

(b) Using this solver, compute the LMPs at each node in the network. You can use numerical differentiation to do this (i.e., solve the optimal power flow problem twice, once with a slightly larger load, and compute the derivative of total cost with respect to this increase numerically). You'll want to explicitly separate out generator power and load power at the generation nodes (you can assume that all load power at these nodes is zero, but you'll still need to use this formulation for computing LMPs at the nodes).