**Project 3**

At the end of this exercise you will be able to:

- Solve a problem using an iterative solution
- Extract a parameter model for a device or system to match given data
- Solve using a system of equations, and find the root solution

Diodes solutions require iterative and nonlinear solvers. Let's first consider the most basic diode problem. We can model a diode with this equation $I = I_s\left(e^{qV/nkT} - 1\right)$ (Eq.1). The complication comes when you place this diode in a circuit. Consider the case where the diode is in series with a resistor R. The current voltage relationship is, of course, Ohm's Law, V=IR. Note that I is on both sides of the equation below, suggesting several ways to solve for it. We will solve for I using the diode equation above and nodal analysis using the circuit below.

$$I = I_s\left(e^{(V-IR)q/nkT} - 1\right)$$

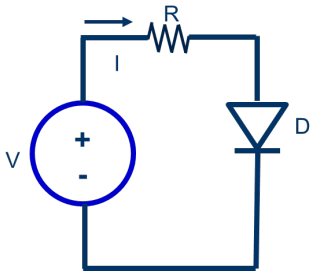
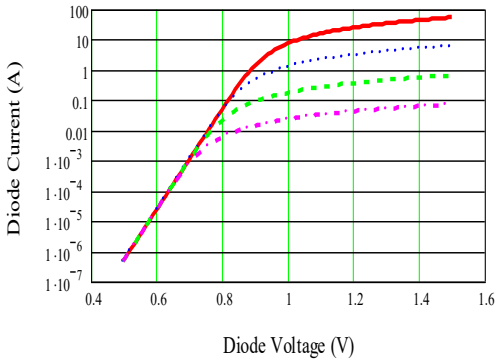



Figure 1




These plots show several different resistances, but your answer will look different. (Notice that the y-axis is a log scale!)

**We will write some code in class to get you started with each of these problems.**

**There are two parts to this project. Both parts should be contained in the SAME python script called project3.py. The two parts of the problem will have functions in common and it is best if both parts use the common functions rather than implementing the same function twice.**

**Problem 1:** Determine the current, I, thorough the circuit in Figure 1 by doing a nodal analysis for the circuit and using the current through the diode defined by the diode equation. Then solve using optimize.fsolve to find the voltage across

the diode. Use these parameters for the diode: Is=1e-9, n=1.7, R=11k ohms, T = 350 and applied voltage V from 0.1 to 2.5 V. Analyze with steps of 0.1 V.

Create one plot with two curves: (a) log(Diode Current) vs Source Voltage; and (b) log(Diode Current) vs Diode Voltage.

Your method is to write the equation using nodal analysis for the node connecting the resistor and the diode, and then write the current equation for the diode in terms of the voltage across it. This should be something like

$Err = \frac{V_d}{R} - \frac{V}{R} + Idiode(V_d)$ , where Err should be zero when the value for $V_d$, the voltage across the diode, is correctly chosen.

**Problem 2:** We are going to repeat the diode problem above for a diode where the parameters are not known. The function for this diode is shown below.

```
def DiodeI(Vd,A,phi,n,T):
    k = 1.380648e-23
    q = 1.6021766208e-19
    Vt = n*k*T/q
    Is = A*T*T*np.exp(-phi*q/(k*T))
    return Is*(np.exp(Vd/Vt)-1)
```

Where phi is the barrier height, T is the temperature, R is the lead resistance, A is the cross-sectional area of the diode, n is the ideality, and Vd is the voltage across the diode.

a) Assume this diode is placed in the circuit in Figure 1 where the lead resistance R is included. Using the methods you developed in problem 1, create a function that returns the currents through this diode for supplied voltage Vs (where Vs is an array of voltages). The returned current should also be an array. In this function you will solve for the current at each supplied voltage as you did in problem 1 and return the array of currents.

b) You are supplied a data set named DiodeIV.txt. The objective is to find the three missing diode parameters. This is a diode which may have uncommon material parameter values. You will be given A, and T, of the measured data and you must determine the other parameters n, phi, and R.

Use scipy optimize leastsq, which will require you optimize one parameter at a time, iteratively, until the solution is achieved. (It may be possible to optimize for all three parameters at once. However, problems like this often require the parameters to be optimized one at a time, so we'll take that approach.)

- from scipy import optimize
- and then use optimize.leastsq( …

Read the DiodeIV.txt. (You must use the data type float64. The read_csv method in pandas defaults to float64, so no worries there. However, if you use numpy's loadtxt method, make sure to set the dtype to float64!) This file contains two columns of data. The first column is the source voltage (that is, the voltage across the diode and resistor) and the second column is the diode current. The Area is 1e-8 and the temperature is 375 Kelvin. Try an initial value of Phi of 0.8, an initial Ideality of 1.5, and an initial Resistor value of 10000 ohms. Your objective is to find the actual values for Phi, Ideality, and the Resistor.

To do this you will have to solve for the voltage across the diode, then calculate the diode current for a set of parameters. Then optimize the values of these parameters to give the same currents as a function of source voltage as those in the file.

You will have to write a function that calculates the residual error. Consider returning absolute or normalized error from the residual function which helps fit the low amplitude parts of the curve as well as the high bias parts. It looks something like this:

Absolute Error = (ynew-ylast)
Normalized Error = (ynew-ylast)/(ynew+ylast+1e-15)

When you use optimize.leastsq, the first parameter will be the residual function and the second will be the guess of the parameter you are trying to find. This is

then the first parameter in that residual function. Since you are optimizing three parameters you may want to write identical residual functions with n, phi, and R as the first parameter to facilitate this process.

Then you could optimize them something like this:
while (err>tolerance and iteration<niter):

    phi = optimize.leastsq(residualp, phi, all the other parameters including n and R)
    n = optimize.leastsq(residualn, n, all the other parameters including n and R)
    R = optimize.leastsq(residualR, R, all the other parameters including n and R)

You may get warnings during the first few iterations. This is due to the inability to drive the residual to 0 when the parameters are still way off. Such warnings are normal and should not be a concern.

Err calculation is up to you. One choice is the difference between the parameters in each pass. When the parameters stop changing you might want to stop. You probably want to limit the number of times through the loop to prevent an infinite loop in the case where your code doesn't converge.

Another way to determine if you should stop is based on the residual values passed back by the last leastsq call in the loop. The residual will be an array of values. You could take the absolute value of each entry, sum the absolute values, and divide by the number of values. This will give you the average size of the errors at each data point. When this value gets small enough, you can stop. (This is how my solution works…)

At the bottom of the while loop described above (or whatever loop you choose to use), print the iteration number and the values of the 3 parameters so that we can track the progress following each iteration. When you have found your values, plot the log(Diode Current) vs source voltage for the data in the file and the predictions from your model on the same plot. You should get an excellent fit for the top part of the curve.

Note that leastsq returns an array, and the first element in the array is the array of optimized values. Therefore, when optimizing a single value, you want to invoke it like this:

```
r_val_op = optimize.leastsq(opt_r,r_val,args=(…))
r_val = r_val_opt[0][0]
```