# ELEC 3908 – Experiment 1: PN Junction Diode Parameter Extraction

## 1 Summary

In this experiment, ORCAD Pspice 17.2 is used to simulate the current-voltage (I-V) characteristics of some generic P-N junction diodes. The data will then be used to extract the physical parameters of these devices. Finally, you will fit a neural-net based model to the data and compare it to the physical model.

## 2 Theory

### 2.1 Physical Diode Model

We have discussed several factors affecting the current-voltage characteristics of PN junction diode. We began with the simplest form of the ideal diode equation given as (1).

$$I_D = I_S(e^{qV_D/kT} - 1) \tag{1}$$

Where $V_D$ is the applied diode terminal voltage (V), $I_D$ is the diode current (A), $I_S$ is the diode saturation current (A) and kT/q is the thermal voltage (V).

Following this, we added the effect of generation-recombination (GR) in the depletion region of the diode. This required including the "ideality factor", n, which alters the slope of the log($I_D$) versus $V_D$ plot as in equation (2).

$$I_D = I_S(e^{qV_D/nKT} - 1) \tag{2}$$

Finally, adding the effect of the series resistance, $R_S$, of the device resulted in equation (3).

$$I_D = I_S(e^{q(V_D - I_D R_S)/nKT} - 1) \tag{3}$$

As discussed in lectures 8 and 9, we can extract the various diode parameters from the measured current versus voltage characteristics of the diodes. To allow these measurements to be made accurately and easily, the lab will be performed using the Cadence ORCAD Pspice 17.2

Consult the instructions located in Appendix A to download a student version of PSpice on your machine. Alternatively you can use your doe account to login and run PSpice on the departmental machines.[1]

### 2.2 Neural Networks and Machine Learning

Although the emphasis in this course is on the physics of electrical devices and therefor physically based models. Increasingly, modeling of complex phenomena is being done using machine learning. These are techniques that typically look at data from many sources and extract relationships within that data. Examples are image processing to extract text; face and speech recognition, and genetic analysis. One particular form of machine learning that analyzes input/output relationships is the use of Neural Networks.

Neural Networks are named after the neurons in the human brain and were initially an attempt to model neural activity. However, they have become a very general technique that has little to do with the human neurology, but still share some similarities to how we function as thinking beings. A Neural Net is made up of layers of so called neurons that can be tasked to classify, solve, and predict different outcomes/solutions. However to get it to the point to solve these problems the Neural Network has to be trained. It has to be given inputs and their respected outputs to that it could recognize an algorithmic pattern to train. Each neuron is connected to all the neurons in its next layer and each connection has a weight associated with it. During the training these weights get calibrated according to the data that is fed through it, with a higher weight meaning it has a higher influence than the other neurons. One can think of the process as summing a series of weighted inputs to produce an output,

$$f(x) = w_1 * I_1(x) + w_2 * I_2(x) + ... + w_n * I_n(x) \tag{4}$$

[1] See the video demo on the CuLearn page of using PSpice to get up to speed on the basic setup.

Where f(x) is the linear function/pattern that is being learned, $w_1, w_2, w_n$ are weight/bias coefficients and $I_{1,2,...n}$ is the inputs given through the training.

A neural network has many different designs, the most common and basic is the feed-forward network in which the neurons fire from left to right (input - output layers). In this design each neurons receives inputs from its left and all the inputs are multiplied by their corresponding weight (calculated from the training phase) and if their sum is greater than the so called threshold value that neuron is "lit up" and its data is sent as an input to all the neurons its connected to.

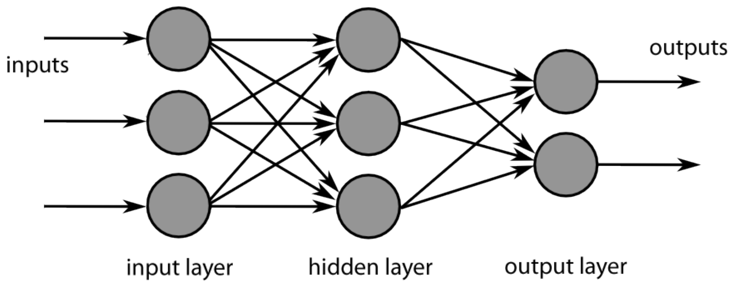


Figure 1: Multi-layer feed forward Neural Network

For the training process the most common method that is used to train these networks is back-propagation, in which the network is given a input, creates an output and takes the difference of the real output and desired output and when the difference is small enough the network is considered trained, if not the cycle repeats.

As neural networks represent an input/output relationship they can be used as a *black box* model of an electrical device relating inputs (typically voltages) to outputs (typically currents). You will build Neural Networks using Matlab that can be used to model diodes and transistors.

## 3 Experiment

You will be simulating[2] a discrete diode and exploring its characteristics under different conditions, as well as furthermore introducing yourself to MATLAB and its powerful Neural Network functionality.

### 3.1 Forward Linear I-V Characteristics

The first measurement will be simple linear $I_D$ versus $V_D$ plots of the diode.

1. Launch ORCAD Capture/PSpice and create a new project, making sure to save it in a known location.

2. Create a basic resistor (1KΩ) diode series circuit with a voltage source powering it using the components located under the PLACE menu, making sure to put a ground in the circuit otherwise the simulation will not run. Also make sure the diode is in the forward bias polarity for this part of the experiment. See figure 2.

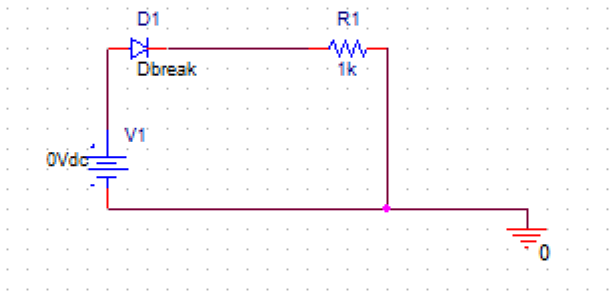
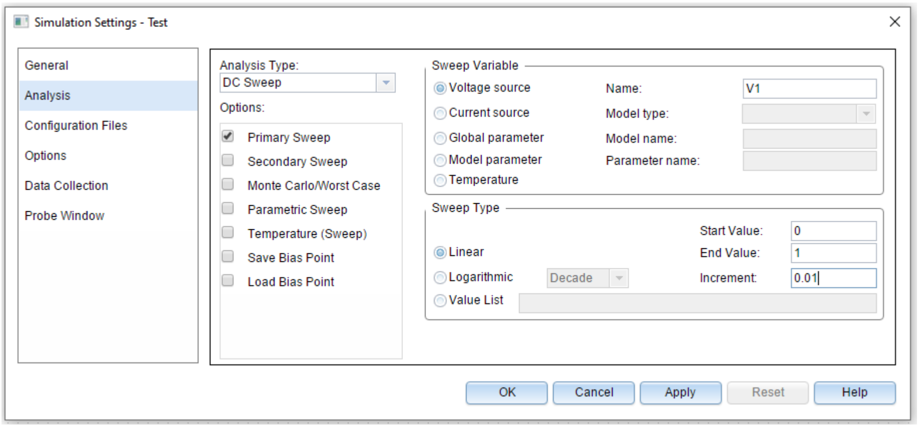


Figure 2: Basic Resistor Diode Circuit

[2]As this is an online lab that replaces a hardware lab we will be simulating measurements. So in the document references to "measured" results refers to simulated results from PSpice.

3. Using the PSPICE menu create a new simulation profile. Change the Analysis Type to DC Sweep. Indicate the voltage source's name (Ex. V1) under the name category. Set the Start Value to 0V and end value of 1V. Use an increment setting of 0.01V. See figure 3.



Simulation Settings - Test

General
Analysis
Configuration Files
Options
Data Collection
Probe Window

Analysis Type: DC Sweep

Options:
[✓] Primary Sweep
[ ] Secondary Sweep
[ ] Monte Carlo/Worst Case
[ ] Parametric Sweep
[ ] Temperature (Sweep)
[ ] Save Bias Point
[ ] Load Bias Point

Sweep Variable
[● Voltage source] Current source Global parameter Model parameter Temperature
Name: V1
Model type:
Model name:
Parameter name:

Sweep Type
[● Linear] Logarithmic Decade Value List
Start Value: 0
End Value: 1
Increment: 0.01

Figure 3: Simulation Settings for Basic Resistor Diode Circuit

4. Run the simulation, in the PLOT menu adjust the x axis to show voltage over the diode. To do this, go to Plot → Axis Settings, and click on change axis variable. In the box at the bottom labled "Trace Expression" put V(D1:1) - V(D1:2). If your diode is named something other than D1, then change the name accordingly. This will display the voltage over the diode.

5. Add a trace of the current across the diode using the TRACE menu option, if done correctly a graph should now appear. While on the graph screen, export your data for later use in MATLAB. This can be done by going to FILE → EXPORT → CSV → Select a name for the file → OK (the file saves in the location of your PSPICE project).

6. In the schematic select the diode and then right click and select "Edit PSpice Model" then change the Is parameter by a factor 3 in the text window, save the new parameters, and re-simulate and save your data.

## 3.2 Forward Logarithmic I-V Characteristics

We will now look at the logarithmic $I_D$ versus $V_D$ plots of the diode.

1. Switch back to the default Diode by re-setting the IS value to 1e-14.

2. Add another y axis to the existing graph (found in the PLOT menu) from 3.1.4 and trace the current again however modify the axis settings (again under the PLOT/AXIS SETTINGS menu) so that the Y Axis scale is set to Log, keep the x axis linear. **Capture the plot for your report**, making sure to label everything correctly.

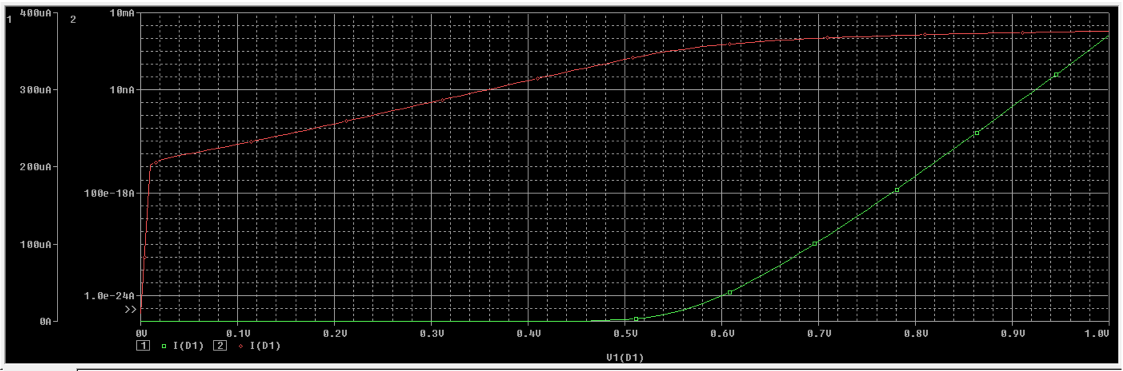


Figure 4: Zener Diode Graph

3. Repeat again with a alteration to the diode IS parameter.

## 3.3 Reverse Linear I-V Characteristics

To measure the reverse characteristics of the diodes we will simply reverse the orientation of the devices so that $V_F$ is now connected to the cathode rather than the anode. We will also need to change the sweep parameters.

1. Switch the polarity of the diode in the circuit by rotating it 180°. Consult Figure (5) for clarification.

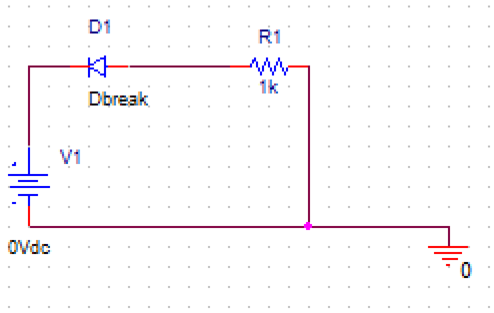


Figure 5: Reverse Bias Diode Circuit

2. In the Simulation Profile change to voltage to sweep from 0 to 20V with a step size of 0.5V.
3. Run the simulation and add a trace of the current over the diode. Modify the graph to have voltage over the diode [V(D1:1) - V(D1:2)] as the X Axis and keep both scale linear.
4. While on the graph screen, export your data for later use in MATLAB.

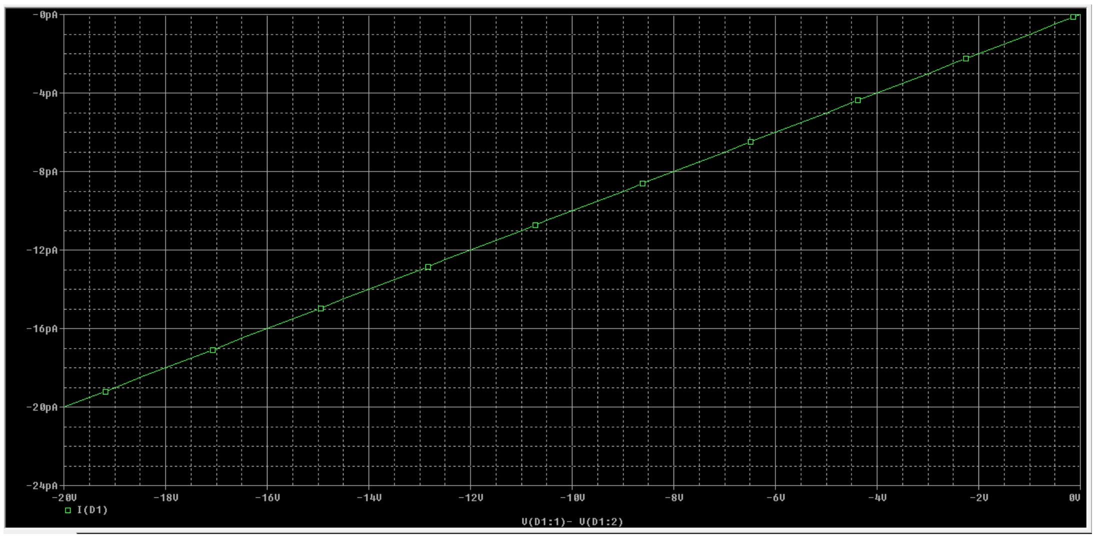


Figure 6: Plot of Reverse Linear Graph

5. Now add a new parameter to the diode model "BV = 16". This sets the breakdown voltage to 16. Plot the results and save them.

## 3.4 Reverse Logarithmic I-V Characteristics

We will now take a look at the logarithmic $I_D$ versus $V_D$ plots of the diodes under reverse bias.

1. Edit the simulation profile and set the start value of the voltage sweep to a number around 0V (ex. 0.01V)

2. Run the simulation, add a trace of current across the diode. Make sure to set the X Axis to voltage across diode. Make the Y Axis a log scale, and **capture the plot for your report**, making sure to label everything correctly.

3. Now again add the parameter to the diode model "BV = 16" (as before select the diode and right click to access "Edit PSpice Model" – however, it may still be open). This sets the breakdown voltage to 16. Plot the results and save them.

### 3.5 Full diode characteristics

Create a circuit to generate the complete diode I-V curve from -3.75V to 0.9V for a diode with the break down voltage set to 3.0V and IS = 1E-14. Use simple a voltage source and diode – no 1K resistor. Save the I-V curve in a CSV file so that you can build a NN model from it.

## 4 Analysis

### 4.1 MATLAB Data Import and Plotting

1. Read the data into MATLAB and recreate the plots you generated with PSpice using the MATLAB plot function. This can be done by creating a New Script and saving it into a know location on the computer. Next Under HOME → IMPORT DATA, navigate to the first .csv file from PART 1 and select it. Change the OUTPUT TYPE from TABLE → NUMERIC MATRIX. Select all but the first row and click Import Selection.

2. Next run the code below and capture the graph from the data that you imported

   CODE (Diode_Data is the name of the numeric table of data that was imported):

```
voltage = Diode_Data(:,3) - Diode_Data(:,4);
current = Diode_Data(:,2);
figure('Name','Measured Diode Data');
plot(voltage, current)
xlabel('Diode Voltage')
ylabel('Diode Current')
```

3. Repeat for all export files collected. Take note that some minor modifications to the code might be required. Especially if your data is in different columns from the example code **Save and capture all of these plots to be included in your write up.**

### 4.2 Parameter Extraction Using MATLAB

Two parameters must be extracted. These are the saturation current, and the ideality factor. To extract these parameters, the graph will be cut, and only the linear part of the data will be analysed. This is due to the fact that this model breaks down at higher voltages. This will only be done for the default diode.

1. First import the table as you did while graphing it.

2. Next look you must decide where to cut the graph. Look at the $\ln(I_D)$ vs $V_D$ that you created in PSpice earlier and select the voltages at which the current diverts from linear(Should be around 0.1V-0.6V). Next you have to go into the text file and find out which rows those voltage appears. Those row numbers are the numbers that you will put instead of StartValue and EndValue in the following code.

```
voltage = Diode_Data(:,3)-Diode_Data(:,4);
logcurrentdata = log(Diode_Data(:,2));
linsectioncurrent = logcurrentdata(StartValue : EndValue);
linsectionvoltage = voltage(StartValue : EndValue);
lincoefficients = polyfit(linsectionvoltage, ...
    linsectioncurrent, 1);
Slope = lincoefficients(1)
Yint = lincoefficients(2)
```

3. This code will extract the values of the slope and the intercept of the linear part of the graph. As long as $V_D >> V_R$, then: $\frac{nkT}{q}$ may be read off directly as the inverse of the slope of the line($\frac{1}{slope}$). Divide this value by 0.0259 ( kT/q at room temperature) to give the value of $n$ . The intercept of the line, given directly from the data as Y intercept, is the value of $ln(I_S)$. Calculate the exponential of this value to give $I_S$ in amps. **Present these values in your report**.

4. Repeat steps 1 - 2 with the data from the diode in reverse bias. What can you extract from this data? For the diode with breakdown what else could you extract?

5. Using the extracted parameters, calculate the diode current, and compare your calculations with the simulation and the neural network in a table.

### 4.3 Forward Linear I-V Characteristics

1. Is there any evidence of series resistance in the forward linear I-V characteristics? If so, how does it manifest itself?

2. Estimate the "turn-on voltage" for the diodes? What would cause the turn-on voltage to differ between different diodes?

### 4.4 Forward Logarithmic I-V Characteristics

1. Is the effect of series resistance in the forward logarithmic I-V characteristics more or less evident than in the linear characteristics?

2. Are the $\log(I_D)$ versus $V_D$ curves linear (for $V_D > \frac{3kT}{q}$) as they would be in an ideal diode? What effects do you think would cause them to deviate from linear behavior

### 4.5 Reverse Linear I-V Characteristics

1. What important characteristics of the diodes were evident from the reverse linear characteristics?

2. What conclusions can be draw between the forward bias data and the reverse bias data values?

### 4.6 Reverse Logarithmic I-V Characteristics

1. The diode equations in the theory section (equations 1,2 and 3) would indicate that we should expect a constant of $I_S$ that is independent of reverse bias voltage. Is this what we see? Give possible reasons for any discrepancies.

2. What are the 5 general types of diodes, there unique characteristics, and what application that they are used in?

### 4.7 Reverse I-V Data Collection

1. Attempt to estimate a reasonable value for $I_S$ for each of the diodes from the collected reverse bias data.

## 5 Neural-Network Model building

This section is for the full diode characteristic you saved in Sec. 3.5.

See the demo in the lecture before the 1st week of labs for information on how to use "nnstart"

1. For this part you need to divide your column data that was imported into individual column vectors. This is already done in the code used to graph the data, so all you need to do is remove the plot function from that code and run it again. You need an input vector of voltages and an output vector of currents.

2. In the terminal type "nnstart", this should bring up a GUI of the Deep Learning Toolbox of MATLAB.

3. Click on “Fitting App”, “Next”, in the Inputs drop down point to your Voltage across the the diode and for Targets select the current column vector.
4. Click the Matrix rows, and then Next
5. Keep all the Validation and Test Data default and click “Next”
6. For the Number of Hidden Neurons keep it at 10 to start (will be modified later), then click “Next”.
7. Using the Levenberg-Marquardt training algorithm, click “Train”, then click “Next”, twice passing the deploy solution screen
8. On the Save Results screen, click ”Simple Script” and save the file on your computer. Run the file, now your net is created. To access it type the command Ex.”net(0.1)”. You may notice that the network is not close to the numbers it should be, to make it more accurate play around with the number of neurons you use (Take into account the more neurons the more computer performance you need).
9. Using the code below you can compare the plot of the data exported from PSPICE and the plot of the Neural Network (Take a look every time you change the number of neurons, it is quite cool).

   CODE (Might need to be slightly changed for each plot):

   ```
   s = 1000
   a = 0.01
   Input =  [ ]
   Output = [ ]
   for c = 1:s
       Input = [Input;a];
       Output = [Output;net(a)];
       a = a +0.01;
       s+1;
   end
   plot(Input,Output)
   ```

10. Compare the output to the original data.
11. Build networks for the other diodes and configurations and test the NN modeling.
12. The code above to plot the Input/Output is not very elegant. There is a way to do this without the loop – how would you do that?