ECE 437 Post-Lab 4

libiano2

September 2023

Post-Lab Questions

1. **Plot the current -voltage characteristic of the diode and submit it with your post-lab report. Include your Python code with the report. (10 pts)**

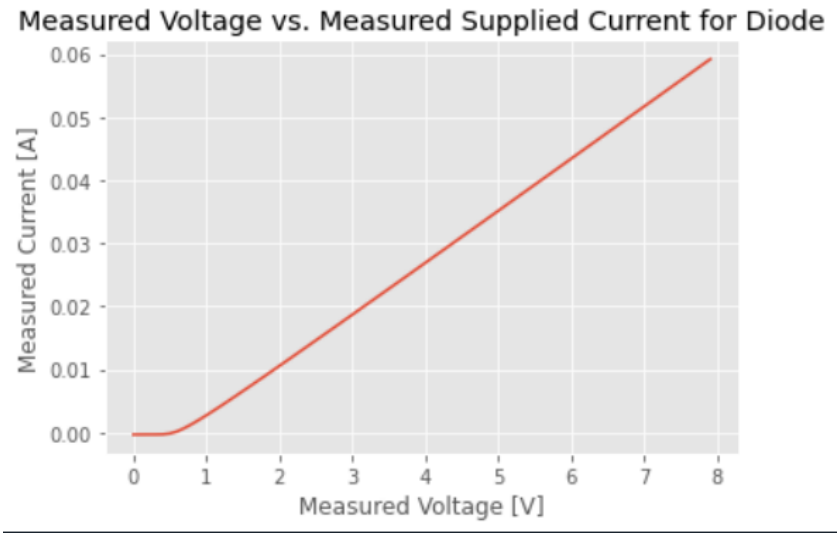

Figure 1: Diode Meaurement Using Step-Size of .1V

```
        # NOTE 1
# If your power supply goes into an error state (i.e., the word
# error is printed on the front of the device), use this command
# power_supply.write("*CLS")
# to clear the error so that you can rerun your code. The supply
# typically beeps after an error has occured.

import pyvisa as visa
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
import time
mpl.style.use('ggplot')

#%%
# This section of the code cycles through all USB connected devices to the computer.
# The code figures out the USB port number for each instrument.
# The port number for each instrument is stored in a variable named \instrument_id"
# If the instrument is turned off or if you are trying to connect to the
# keyboard or mouse, you will get a message that you cannot connect on that port.
device_manager = visa.ResourceManager()
devices = device_manager.list_resources()
number_of_device = len(devices)

power_supply_id = -1;
waveform_generator_id = -1;
digital_multimeter_id = -1;
oscilloscope_id = -1;

# assumes only the DC power supply is connected
for i in range (0, number_of_device):

# check that it is actually the power supply
    try:
        device_temp = device_manager.open_resource(devices[i])
        print("Instrument connect on USB port number [" + str(i) + "] is " + device_temp.que
        if (device_temp.query("*IDN?") == 'HEWLETT-PACKARD,E3631A,0,3.2-6.0-2.0\r\n'):
            power_supply_id = i
        if (device_temp.query("*IDN?") == 'HEWLETT-PACKARD,E3631A,0,3.0-6.0-2.0\r\n'):
            power_supply_id = i
        if (device_temp.query("*IDN?") == 'Agilent Technologies,33511B,MY52301259,3.03-1.19-
            waveform_generator_id = i
        if (device_temp.query("*IDN?") == 'Agilent Technologies,34461A,MY53207926,A.01.10-02
            digital_multimeter_id = i
        if (device_temp.query("*IDN?") == 'Keysight Technologies,34461A,MY53212931,A.02.08-0
            digital_multimeter_id = i
        if (device_temp.query("*IDN?") == 'KEYSIGHT TECHNOLOGIES,MSO-X 3024T,MY54440281,07.1
            oscilloscope_id = i
        device_temp.close()
    except:
        print("Instrument on USB port number [" + str(i) + "] cannot be connected. The instr


#%%
# Open the USB communication port with the power supply.
# The power supply is connected on USB port number power_supply_id.
# If the power supply ss not connected or turned off, the program will exit.
# Otherwise, the power_supply variable is the handler to the power supply

if (power_supply_id == -1):
```

```
    print("Power supply instrument is not powered on or connected to the PC.")
else:
    print("Power supply is connected to the PC.")
    power_supply = device_manager.open_resource(devices[power_supply_id])

#%%
# The power supply output voltage will be swept from 0 to 1.5V in steps of 0.05V.
# This voltage will be applied on the 6V output ports.
# For each voltage applied on the 6V power supply, we will measure the actual
# voltage and current supplied by the power supply.
# If your circuit operates correctly, the applied and measured voltage will be the same.
# If the power supply reaches its maximum allowed current,
# then the applied voltage will not be the same as the measured voltage.

    output_voltage = np.arange(0, 8, 1)
    measured_voltage = np.array([]) # create an empty list to hold our values
    measured_current = np.array([]) # create an empty list to hold our values

    print(power_supply.write("OUTPUT ON")) # power supply output is turned on

    # loop through the different voltages we will apply to the power supply
    # For each voltage applied on the power supply,
    # measure the voltage and current supplied by the 6V power supply
    for v in output_voltage:
        # apply the desired voltage on teh 6V power supply and limist the output current to
        power_supply.write("APPLy P25V, %0.2f, 0.06" % v)

        # pause 50ms to let things settle
        time.sleep(0.5)

        # read the output voltage on the 6V power supply
        measured_voltage_tmp = power_supply.query("MEASure:VOLTage:DC? P25V")
        measured_voltage = np.append(measured_voltage, measured_voltage_tmp)

        # read the output current on the 6V power supply
        measured_current_tmp = power_supply.query("MEASure:CURRent:DC? P25V")
        measured_current = np.append(measured_current, measured_current_tmp)

    # power supply output is turned off
    print(power_supply.write("OUTPUT OFF"))

    # close the power supply USB handler.
    # Otherwise you cannot connect to it in the future
    power_supply.close()

#%% Plot measured data. First convert the data from strings to numbers (ie floats)
    voltage_list=np.zeros(np.size(output_voltage))
    current_list=np.zeros(np.size(output_voltage))
    for i in range(len(measured_voltage)):
```

```
        voltage_list[i]= float(measured_voltage [i])
        current_list[i]= float(measured_current[i])

    # plot results (applied voltage vs measured supplied current)
    plt.figure()
    plt.plot(output_voltage, current_list)
    plt.title("Applied Volts vs. Measured Supplied Current for Diode")
    plt.xlabel("Applied Volts [V]")
    plt.ylabel("Measured Current [A]")
    plt.draw()
    # plot results (measured voltage vs measured supplied current)
    plt.figure()
    plt.plot(voltage_list, current_list)
    plt.title("Measured Voltage vs. Measured Supplied Current for Diode")
    plt.xlabel("Measured Voltage [V]")
    plt.ylabel("Measured Current [A]")
    plt.draw()
    # show all plots
    plt.show()
```

2. **Change the increment step size to 1 V and comment on the difference between the results obtained in question Q1. Include a plot of the current-voltage characteristics of the diode with the new increment step size. (10 pts)**

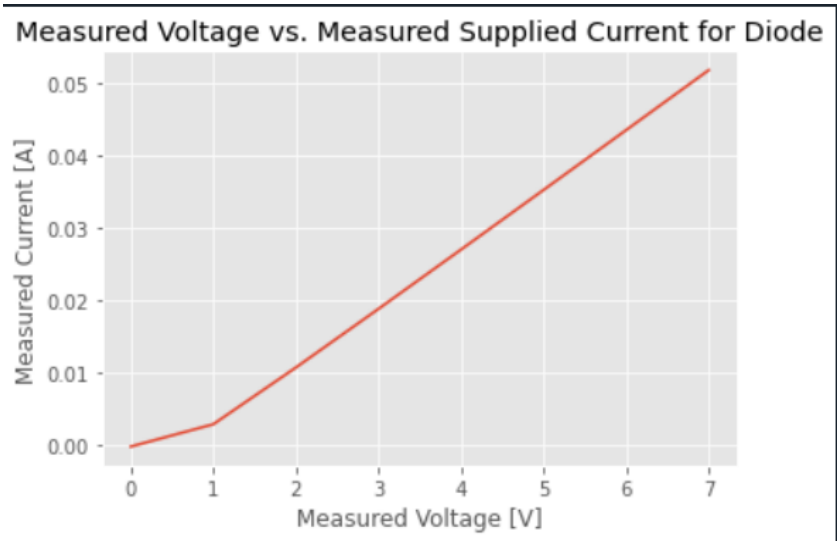

Figure 2: Diode Meaurement Using Step-Size of 1V

Using a step-size of 1V, we see that we our plot becomes less precise in measurement. Since we are limiting the amount of data points we can use in the measurement, we see that instead of looking like a non-linear function the measurement looks like it is two piece

wise linear functions. Compared to our original measurement with step-size of .1V, our graph is not an accurate representation of what we expect for a diode measurement.

3. **What would be a reasonable step function when you are plotting the I-V characteristics of a device and how will you determine it? What are some of the tradeoffs when you are determining the increment size? (10 pts)** Ideally, we found that a linear step function in the form of $V_{out} = V_{step} * n + D.C.offset$ where n is the current step was the best way to measure the I-V characteristics of the diode. Furthermore, we find that choosing $V_{step} << D.C.offset$ gave us the best looking I-V plot for the diode. Some tradeoff's to consider, though, is the time it will take for the program to step (i.e. having a small stepsize is great for precision, but not great if you are bounded by a time limit in a testing scenario), the size of memory or storage on the pc (a smaller step size will also generate a lot of data that the test system may not be able to process), and the granularity of the measurements (as we saw above, larger stepsizes are less precise versus smaller step sizes).

4. **Print both your Python code and the output results from running this code and include them in your lab report. Comment on the results you have obtained from this checkpoint. Why is this current-voltage characteristic different compared to the one you collected in the first checkpoint? (60 pts)**

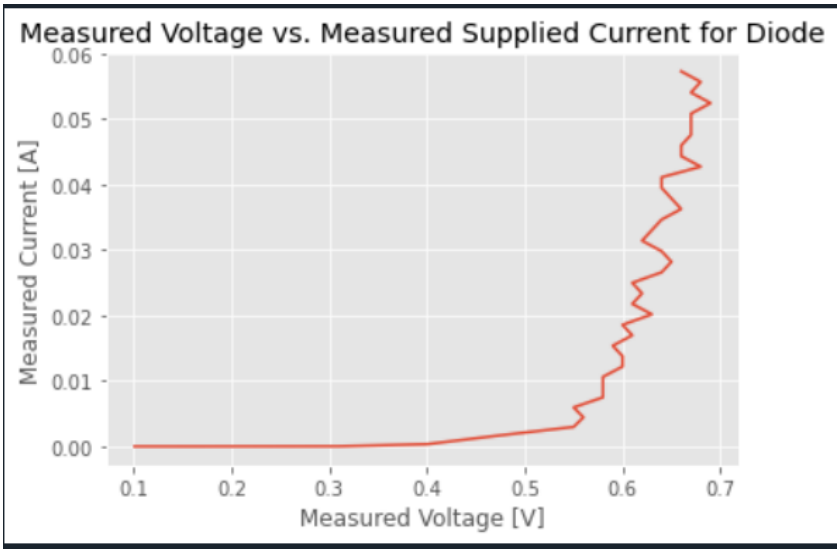

Figure 3: Diode Meaurement Using a DMM, Power Supply and Oscilloscope

The current-voltage characteristic looks different than the one collected at the first checkpoint since the measurements across the diode are not as clean with the oscilloscope and ammeter setup as compared to using the built in ammeter and voltmeter of the power-supply. When the power-supply steps to a certain voltage, the current is automatically stepped up as well to provide enough current to the connected device. This means that both the voltage and current are coherent in measurement since the measurement is taken from the same device. With the DMM as an ammeter and the oscilloscope as a voltmeter, both measurements are not completely or exactly coherent. In our code, current is mea-

sured first and the voltage. A way to circumvent this is to find a way to have both the ammeter and voltmeter in our setup take one-shot at the same time for each time step. We also noticed with the oscilloscope that the readings were not a straight line across every time there was a new time step. Around the 4V-5V input range from the power supply, we see the oscilloscope measures some sort of periodic wave as opposed to a straight DC line like we expect. To circumvent this problem, we can place an AC blocking circuit in the measurement setup to only measure the DC component.

```
        # NOTE 1
# If your power supply goes into an error state (i.e., the word
# error is printed on the front of the device), use this command
# power_supply.write("*CLS")
# to clear the error so that you can rerun your code. The supply
# typically beeps after an error has occured.

import pyvisa as visa
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import time
mpl.style.use('ggplot')

#%%
# This section of the code cycles through all USB connected devices to the computer.
# The code figures out the USB port number for each instrument.
# The port number for each instrument is stored in a variable named \instrument_id"
# If the instrument is turned off or if you are trying to connect to the
# keyboard or mouse, you will get a message that you cannot connect on that port.
device_manager = visa.ResourceManager()
devices = device_manager.list_resources()
number_of_device = len(devices)

power_supply_id = -1;
waveform_generator_id = -1;
digital_multimeter_id = -1;
oscilloscope_id = -1;

# assumes only the DC power supply is connected
for i in range (0, number_of_device):

# check that it is actually the power supply
    try:
        device_temp = device_manager.open_resource(devices[i])
        print("Instrument connect on USB port number [" + str(i) + "] is " + device_temp.que
        if (device_temp.query("*IDN?") == 'HEWLETT-PACKARD,E3631A,0,3.2-6.0-2.0\r\n'):
            power_supply_id = i
        if (device_temp.query("*IDN?") == 'HEWLETT-PACKARD,E3631A,0,3.0-6.0-2.0\r\n'):
            power_supply_id = i
        if (device_temp.query("*IDN?") == 'Agilent Technologies,33511B,MY52301259,3.03-1.19-
```

```
            waveform_generator_id = i
        if (device_temp.query("*IDN?") == 'Agilent Technologies,34461A,MY53207929,A.01.10-02
            digital_multimeter_id = i
        if (device_temp.query("*IDN?") == 'Keysight Technologies,34461A,MY53212931,A.02.08-0
            digital_multimeter_id = i
        if (device_temp.query("*IDN?") == 'KEYSIGHT TECHNOLOGIES,MSO-X 3024T,MY55100339,07.5
            oscilloscope_id = i
        device_temp.close()
    except:
        print("Instrument on USB port number [" + str(i) + "] cannot be connected. The instr


#%%
# Open the USB communication port with the power supply.
# The power supply is connected on USB port number power_supply_id.
# If the power supply ss not connected or turned off, the program will exit.
# Otherwise, the power_supply variable is the handler to the power supply
if (power_supply_id == -1):
    print("Power supply instrument is not powered on or connected to the PC.")
else:
    print("Power supply is connected to the PC.")
    power_supply = device_manager.open_resource(devices[power_supply_id])

if (digital_multimeter_id == -1):
    print("DVM is not powered on or connected to the PC.")
else:
    print("DVM is connected to the PC.")
    digital_multimeter = device_manager.open_resource(devices[digital_multimeter_id])

if (oscilloscope_id == -1):
    print("Osco instrument is not powered on or connected to the PC.")
else:
    print("Osco supply is connected to the PC.")
    oscilloscope = device_manager.open_resource(devices[oscilloscope_id])

#%%
# The power supply output voltage will be swept from 0 to 1.5V in steps of 0.05V.
# This voltage will be applied on the 6V output ports.
# For each voltage applied on the 6V power supply, we will measure the actual
# voltage and current supplied by the power supply.
# If your circuit operates correctly, the applied and measured voltage will be the same.
# If the power supply reaches its maximum allowed current,
# then the applied voltage will not be the same as the measured voltage.

    output_voltage = np.arange(0, 8.0, .2)
    measured_voltage = np.array([]) # create an empty list to hold our values
    measured_current = np.array([]) # create an empty list to hold our values

    print(power_supply.write("OUTPUT ON")) # power supply output is turned on
```

```
    # loop through the different voltages we will apply to the power supply
    # For each voltage applied on the power supply,
    # measure the voltage and current supplied by the 6V power supply
    for v in output_voltage:
        # apply the desired voltage on teh 6V power supply and limist the output current to
        power_supply.write("APPLy P25V, %0.2f, 0.06" % v)

        # pause 50ms to let things settle
        time.sleep(.5)

        # read the output voltage on the 6V power supply
        measured_voltage_tmp = oscilloscope.query("MEASure:VRMS? DISPlay,DC,CHANnel2")
        measured_voltage = np.append(measured_voltage, measured_voltage_tmp)

        # read the output current on the 6V power supply
        measured_current_tmp = digital_multimeter.query("MEASure:CURRent:DC?")
        measured_current = np.append(measured_current, measured_current_tmp)

    # power supply output is turned off
    print(power_supply.write("OUTPUT OFF"))

    # close the power supply USB handler.
    # Otherwise you cannot connect to it in the future
    power_supply.close()
    oscilloscope.close()
    digital_multimeter.close()

#%% Plot measured data. First convert the data from strings to numbers (ie floats)
    voltage_list=np.zeros(np.size(output_voltage))
    current_list=np.zeros(np.size(output_voltage))
    for i in range(len(measured_voltage)):
        voltage_list[i]= float(measured_voltage[i])
        print(voltage_list[i])
        current_list[i]= float(measured_current[i])
        print(current_list[i])

    # plot results (applied voltage vs measured supplied current)
    plt.figure()
    plt.plot(output_voltage, current_list)
    plt.title("Applied Volts vs. Measured Supplied Current for Diode")
    plt.xlabel("Applied Volts [V]")
    plt.ylabel("Measured Current [A]")
    plt.draw()
    # plot results (measured voltage vs measured supplied current)
    plt.figure()
    plt.plot(voltage_list, current_list)
    plt.title("Measured Voltage vs. Measured Supplied Current for Diode")
    plt.xlabel("Measured Voltage [V]")
    plt.ylabel("Measured Current [A]")
```

```
plt.draw()
# show all plots
plt.show()
```