**Photoelectric Effect Python**

Go to [https://applets.kcvs.ca/photoelectricEffect/PhotoElectric.html#](https://applets.kcvs.ca/photoelectricEffect/PhotoElectric.html)

To start your simulation experiment to collect data to use in your Python code. In the options menu on the upper left-hand corner, select a metal to analyze. You can choose from Cs, Ca, K, or Na.

Adjust the wavelength to 200 nm. Record the frequency. Adjust the voltage by 0.25 and click record the data point. You can stop recording data once you have reached 5 V, click show data and download the CSV. Save the CSV file according to the name of the metal you are analyzing and the wavelength. Use an underscore instead of a space to save the file (Example Cs\_200nm.csv). Go to the upper left-hand corner under options and clear the data. Start the process again at 250, 300, 350, 400, 450, and 500 nm. Put these files in your Python folder in the Photoelectric effect experiment on your desktop.

**Question 1**

Write a python code that will plot one data file and include a polynomial fit. Include correct data labels and the overlay the polynomial fit on your graph. Display the key for your experiment data and the polynomial fit. Include the equation on your graph. Export the graphs a PNG to put into your Post-Lab analysis.

1. def plot\_data\_with\_fit(file\_path, x\_column, y\_column, poly\_degree=2, output\_file=None): # Added output\_file parameter

2. # Import the CSV data

3. data = import\_csv(file\_path)

4.

5. # Extract the x and y data

6. x = data[x\_column]

7. y = data[y\_column]

8.

9. # Perform polynomial fitting

10. poly\_coeff = np.polyfit(x, y, poly\_degree)

11. poly = np.poly1d(poly\_coeff)

12.

13. # Generate points for plotting the polynomial fit

14. x\_fit = np.linspace(x.min(), x.max(), 500)

15. y\_fit = poly(x\_fit)

16.

17. # Plot the original data

18. plt.scatter(x, y, label='Data Points', color='blue')

19.

20. # Plot the polynomial fit

21. plt.plot(x\_fit, y\_fit, label=f'Polynomial Fit (Degree {poly\_degree})', color='red')

22.

23. # Add labels and title

24. plt.xlabel(x\_column)

25. plt.ylabel(y\_column)

26. plt.title(f'{y\_column} vs {x\_column} for Ca Metal at 200nm')

27.

28. # Format the polynomial equation with LaTeX and superscripts

29. equation\_text = "$y = " + " + ".join([f"{round(coef, 3)}x^{poly\_degree - i}" if i != poly\_degree else f"{round(coef, 3)}"

30. for i, coef in enumerate(poly\_coeff)]) + "$"

31.

32. # Align the text to the center of the chart

33. plt.text(0.4, 0.8, equation\_text, transform=plt.gca().transAxes, fontsize=8,

34. verticalalignment='bottom', horizontalalignment='left')

35.

36. # Show legend and plot

37. plt.legend()

38.

39. # Save the plot as a PNG file if output\_file is provided

40. if output\_file: # Check if output\_file is not None

41. plt.savefig(output\_file, format='png', dpi=300, bbox\_inches='tight') # DPI = 300 for high quality

42.

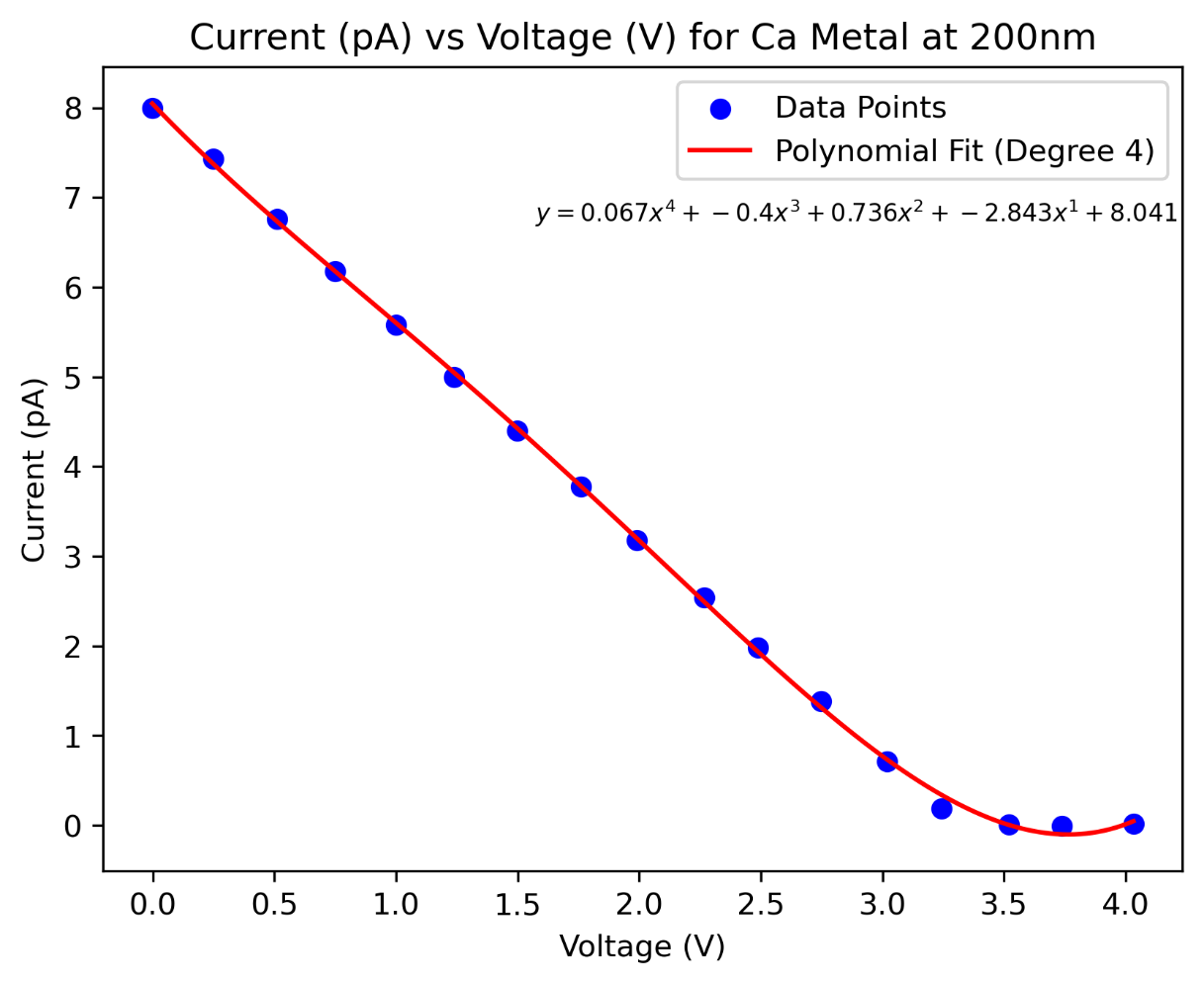
43. plt.show()

44.

45. # Example usage

46. file\_path = 'Ca\_200nm.csv' # Replace with your CSV file path

47. plot\_data\_with\_fit(file\_path, 'Voltage (V)', 'Current (pA)', poly\_degree=4, output\_file='Ca\_200nm.png')



**Question 2**

Ask the code to calculate the y and x intercepts for the polynomial fit. Take a screenshot of the output and include it in your post-run file.

**Question 3**

Write a python code that will plot all the same data file with a linear regression fit. Include correct data labels and overlay the linear fit on your graph. Display the key and the equation on your graph. Export the graphs a PNG to put into your Post-Lab analysis.

1. def plot\_data\_with\_fit(file\_path, x\_column, y\_column, output\_file=None): # Removed poly\_degree parameter for linear fit

2. # Import the CSV data

3. data = import\_csv(file\_path)

4.

5. # Include only the first 15 rows (0-14)

6. data = data.iloc[0:15] # Slicing the DataFrame to keep only the first 15 rows

7.

8. # Filter the data to remove points where Voltage exceeds 3.520 volts

9. data = data[data[x\_column] <= 3.520] # Keep only rows where Voltage (x\_column) is less than or equal to 3.520 volts

10.

11. # Extract the x and y data

12. x = data[x\_column]

13. y = data[y\_column]

14.

15. # Perform linear fitting (degree 1)

16. slope, intercept = np.polyfit(x, y, 1) # Fit a line (linear regression)

17.

18. # Calculate the x-intercept

19. if slope != 0: # Avoid division by zero

20. x\_intercept = -intercept / slope # x-intercept when y = 0

21. else:

22. x\_intercept = None # If slope is zero, x-intercept is undefined

23.

24. # Generate points for plotting the linear fit

25. x\_fit = np.linspace(x.min(), x.max(), 500)

26. y\_fit = slope \* x\_fit + intercept # Calculate y values based on the linear equation

27.

28. # Plot the original data

29. plt.scatter(x, y, label='Data Points', color='blue')

30.

31. # Plot the linear fit

32. plt.plot(x\_fit, y\_fit, label='Linear Fit', color='red')

33.

34. # Add labels and title

35. plt.xlabel(x\_column)

36. plt.ylabel(y\_column)

37. plt.title(f'{y\_column} vs {x\_column} for Ca Metal at 200nm')

38.

39. # Format the linear equation for display

40. equation\_text = f"$y = {round(slope, 3)}x + {round(intercept, 3)}$"

41.

42. # Align the text to the center of the chart

43. plt.text(0.4, 0.8, equation\_text, transform=plt.gca().transAxes, fontsize=8,

44. verticalalignment='bottom', horizontalalignment='left')

45.

46. # Show legend and plot

47. plt.legend()

48.

49. # Save the plot as a PNG file if output\_file is provided

50. if output\_file: # Check if output\_file is not None

51. plt.savefig(output\_file, format='png', dpi=300, bbox\_inches='tight') # DPI = 300 for high quality

52.

53. plt.show()

54.

55. return slope, intercept, x\_intercept # Return the slope, intercept, and x-intercept of the linear fit

56.

57. # Example usage

58. file\_path = 'Ca\_200nm.csv' # Replace with your CSV file path

59. slope, intercept, x\_intercept = plot\_data\_with\_fit(file\_path, 'Voltage (V)', 'Current (pA)', output\_file='Ca\_200nm.png')

60.

61. # Display the y-intercept and x-intercept

62. print(f"The y-intercept is: {intercept}")

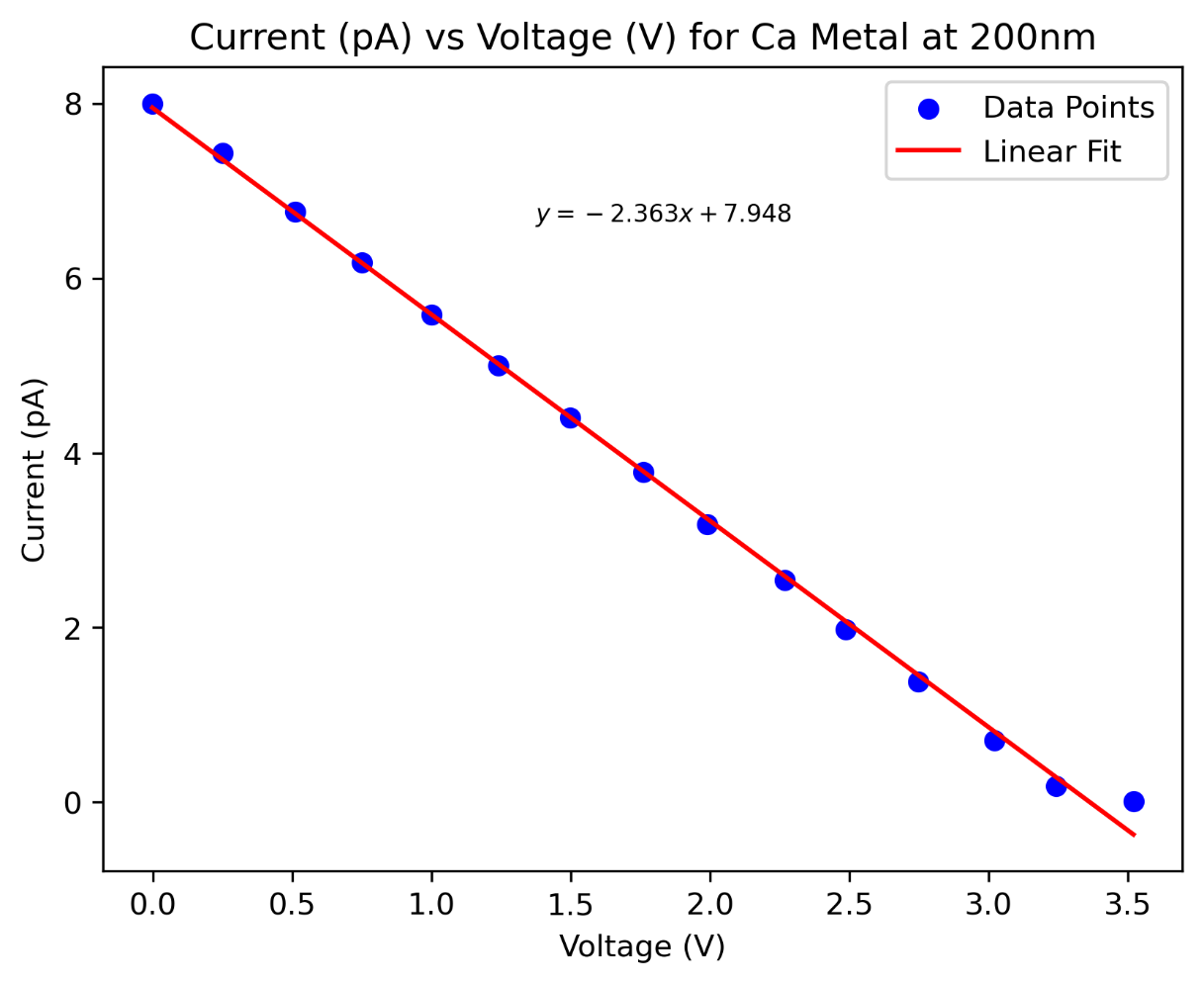
63. if x\_intercept is not None:

64. print(f"The x-intercept is: {x\_intercept} (This is the value of x when y = 0)")

65. else:

66. print("The slope is zero, so the x-intercept is undefined (the line is horizontal).")

67.



**Question 4**

Ask the code to calculate the y and x intercepts for the linear regression fit. Take a screenshot of the output and include it in your post-run file.

**Question 5**

The stopping potential (VO) is equal to your x-intercept. Knowing this you can calculate your work function (*φ*) using the equation

Write a code in python to calculate the work function for both the polynomial fit and the linear regression. Insert the code and output here.

**Question 6**

The cut-off frequency and wavelength can be found using the equation

Write a code in python to calculate the cut off frequency and wavelength for both the polynomial fit and the linear regression. Insert the code and output here.

**Question 7**

Prepare a table for your metal that is similar to the following, change the name “Metal” to represent the metal that you have chosen.

Theoretical Values for “Metal” Using Polynomial Fit

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| λ (nm) | Frequency (Hz) | Stopping Potential (V0) | Work Function (J) | Cut-off Frequency (Hz) | Cut-off Wavelength  (nm) |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Theoretical Values for “Metal” Using Linear Regression

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Λ (nm) | Frequency (Hz) | Stopping Potential (V0) | Work Function (J) | Cut-off Frequency (Hz) | Cut-off Wavelength  (nm) |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Question 8**

Create a plot in python of the stopping potential vs frequency. Include data from your polynomial fit and your linear regression on the same graph. Fit both sets of data with separate linear regression equations. Display these equations in the key with their associated fit.

**Question 9**

Find the experimental Planck’s constant for the polynomial fit and the linear regression using python. Compare this to the literature value and calculate the percent error using python. Upload this code and the output here.

**Question 10**

Which method of fitting is better? Why do you think so?