**סדנת עיבוד נתונים**

בסדנה זו תלמדו לעבד את נתוני המעבדה ב**עזרת שפת Python**. אנחנו נשתמש בתוכנה Spyder לכתיבה והרצה של הקוד. המטרה היא ליצור מיומנות בסיסית של ניתוח נתונים והצגתם בגרפים כדי שתוכלו לעבד את הנתונים של המעבדה בקלות. בנוסף, הקוד שתכתבו בסדנה יישמר אצלכם ותוכלו לחזור ולהיעזר בו בעתיד.

**היקף הסדנה** הוא **6 שעות – 3 בכיתה ו-3 בבית**, בסופן יהיו ברשותכם קבצי קוד שיכילו את כל הפקודות שלמדתם ותוכלו לעשות בהם שימוש במהלך המעבדה. במפגש בכיתה נגיע עד וכולל ניסוי מיפוי פוטנציאל.

**הערה:** סדנה זו כוללת שני חלקים לתרגול: חלקים ד' ו-ה'. סטודנטים אשר עברו את סדנת הפייתון במעבדה 2, רשאים לדלג על חלק ד' ולהגיש רק את חלק ה'.

Contents

[חלק א' – התקנת Anaconda ו-Spyder 1](#_Toc117414399)

[חלק ב' – סביבת עבודה ב-Spyder 2](#_Toc117414400)

[חלק ג' – שימוש בסיסי ב-Python 2](#_Toc117414401)

[חלק ד – עיבוד נתוני המעבדה 3](#_Toc117414402)

[תרגיל - מיפוי פוטנציאל חשמלי 4](#_Toc117414403)

[תרגיל - קבל לוחות 6](#_Toc117414404)

[תרגיל – חוק אוהם 10](#_Toc117414405)

[תרגיל - השראות 11](#_Toc117414406)

[חלק ה' – עיבוד נתונים מעבדת גלים 14](#_Toc117414407)

[תרגיל – ספקטרוסקופיה 14](#_Toc117414408)

[תרגיל – מדידת קיטוב של גלי מיקרו 16](#_Toc117414409)

[תרגיל – האפקט האקוסטו-אופטי 17](#_Toc117414410)

[תרגיל – שני זמזמים בצינור 19](#_Toc117414411)

# חלק א' – התקנת Anaconda ו-Spyder

**בצעו** התקנה של **Anaconda**:

<https://docs.anaconda.com/anaconda/install/windows/>

**התקנה זו מתקינה גם Spyder**.

# חלק ב' – סביבת עבודה ב-Spyder

הערכת זמן: כ- 15 דק'

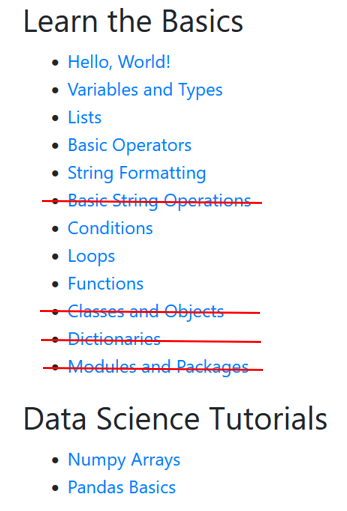
1. **צרו** תיקית קבצים ריקה שתוקדש לסדנה.
2. **פתחו** את Spyder.
3. הסתכלו על הממשק של Spyder **וזהו** את החלקים הבאים:
   1. סרגל פקודות ותפריטים, תיקיית ההרצה.
   2. חלון הקובץ – אזור כתיבת הקוד.
   3. **ה-Console**, שימו לב שישנם **שני Tabs** לחלון זה.
   4. **חלון עזרה**, שימו לב שישנם **4 tabs: Help, Variable Explorer, Plots, Files**
4. **שנו** את תיקיית ההרצה לתיקיית הסדנה – כדי שהקוד יוכל למצוא את הקבצים שנמצאים בה.
5. **צרו קובץ חדש** **tutorial.py** ושמרו אותו בתיקיה שתוקדש לסדנה.
6. **עיברו** עם העכבר מעל האייקונים של פקודות ההרצה

Graphical user interface, application, Teams

Description automatically generated

ו**קראו** את קיצורי המקשים של הרצת קוד ב-Spyder.

# חלק ג' – שימוש בסיסי ב-Python

הערכת זמן: כ-45 דקות

**היכנסו** לאתר <https://www.learnpython.org/en/Welcome>.

כאן נלמד את השיעורים המוגדרים *כלימודי הבסיס וניתוח נתונים* -

אנחנו נבצע את השיעורים עד וכולל Pandas, לפי הרשימה (שימו לב, **מדלגים** על חלק מהשיעורים). אנחנו נקדיש לכך כ-45 דקות. במידה ויש צורך תוכלו להמשיך בבית.

דגשים חשובים (מאוד):

1. במקום להריץ באתר – **העתיקו** את הקוד לקובץ ב-Spyder ו**הריצו** את הפקודות שם. באופן זה תשמרו את כל מה שלמדתם.
2. אם אתם מבינים את דוגמת הקוד, אתם לא חייבים להריץ אותה. תעתיקו ותריצו רק מה שאתם רוצים לוודא שאתם מבינים.
3. **התמקדו** בלהבין את הדוגמאות (ופחות בהבנת הטקסט).
4. **בצעו** את התרגיל בסוף כל שיעור באתר (ב-Spyder).
5. מי שמכיר את כל התכנים האלו – שיעבור לבצע את חלק ד' של הסדנה. אין צורך באישור המדריך.

\*לאחר הסדנה, אתם מוזמנים לבצע את שאר השיעורים אם אתם רוצים.

**שמרו** את הקוד שכתבתם בתיקיה.

הלינק אצלכם, הקוד אצלכם, תוכלו להמשיך ולהתקדם בכל זמן שתרצו, **הכול זמין עבורכם**.

# חלק ד – עיבוד נתוני המעבדה

באופן כללי, הפעולות הנדרשות לעיבוד הנתונים הן:

1. כתיבת וקטור נתונים (למדתם בשיעור Numpy Arrays)
2. חישוב עקום תיאורטי
3. טעינת נתונים מקובץ
4. הצגת נתונים בגרף
5. ביצוע חישוב על וקטורי נתונים (כתיבת פונקציה)
6. ביצוע רגרסיה לינארית
7. ביצוע fit (לעקום לא לינארי)
8. ביצוע אינטגרציה לווקטור נתונים

בחלק זה של הסדנה תלמדו ותבצעו את כל הפעולות האלה, על נתונים מניסויי המעבדה.

כדי לעבד את הנתונים, נשתמש בmodules הבאים:

iimport numpy as np # math functions

import scipy # scientific functions

import matplotlib.pyplot as plt # for plotting figures and setting their properties

import pandas as pd # handling data structures (loaded from files)

from scipy.stats import linregress # contains linregress (for linear regression)

from scipy.optimize import curve\_fit as cfit # non-linear curve fitting

from sklearn.metrics import r2\_score # import function that calculates R^2 score

1. **פתחו** קובץ קוד חדש בשם **data\_analysis.py**
2. **העתיקו** את השורות האלה לתחילת הקובץ של הקוד **והריצו** אותן.

נתחיל לעבור יחד על ניסוי המעבדה באמצעות רקע קצר על מנת שתוכלו לנתח את הנתונים

## תרגיל - מיפוי פוטנציאל חשמלי

**רקע תאורטי**

בניסוי מיפוי פוטנציאל חשמלי אתם תמדדו את הפוטנציאל החשמלי על דף מוליך שמחוברות אליו אלקטרודות. בתרגיל הזה אנחנו נשרטט את הפוטנציאל החשמלי התיאורטי של שתי אלקטרודות ונחשב במפורש את שינוי הפוטנציאל מאלקטרודה אחת לשנייה.

נניח שישנן שתי אלקטרודות (חיובית ושלילית), בנקודות ו-. הפוטנציאל מסביב לאלקטרודה בודדת הוא:

*כאשר מתקיים (מהגדרה זו) . הפוטנציאל משתי אלקטרודות הוא סכום הפוטנציאלים מכל אחת מהן. נבחר את להיות ראשית הצירים, איפה ש- ונקבל*

***המשימה*** *- נשרטט את הפוטנציאל.*

1. ***פתחו*** *cell חדש בקוד (#%%) בשם potential*
2. ***העתיקו*** *את השורות הבאות:*

C = 1

a = 1

L = 3

N = 100

coord = np.linspace(-L, L , N) # defines coordinates

coord\_x, coord\_y = np.meshgrid(coord, coord)

1. ***הריצו*** *את הקוד ו****הסתכלו*** *ב-Variable Explorer על משתני הקואורדינטות (coord\_x, coord\_y, coord). מה הפונקציה linspace ביצעה? מה הפונקציה meshgrid ביצעה (ניתן להסתכל בתיעוד/הסברים באינטרנט)?*
2. ***כתבו*** *פונקציה שמחשבת את הפוטנציאל: def potential(x, y, a, C). (****היעזרו*** *ב: np.log(), np.sqrt(), np.power() לפי הצורך)*
3. ***חשבו*** *את הפוטנציאל בקואורדינטות הנתונות* ***וציירו*** *אותו באמצעות השורות הבאות:*

V\_xy = potential(coord\_x, coord\_y, a, C)

plt.figure()

plt.pcolormesh(coord\_x, coord\_y, V\_xy)

plt.colorbar()

1. ***הריצו*** *את הקוד* ***והוסיפו*** *הערות לכל שורת קוד מסעיף 5 (עם #) שמסבירות מה כל שורה עושה.*
2. *כעת נוסיף לגרף קווים שווי-פוטנציאל, בהם הפוטנציאל הוא בעל ערך קבוע. כלומר, כמו במפות טופוגרפיות בהן מסומנים קווים שווי גובה, נרצה לצייר על הגרף קווים בהם ערך הפוטנציאל קבוע. לשם כך נשתמש בפונקציה contour שמבצעת את החישוב הזה ומציירת קווים בערכי הפוטנציאל שנגדיר לה. contour מקבלת את נתוני הפוטנציאל -x,y,V(x,y)- ואת הרמות levels של הפוטנציאל שאנו רוצים לצייר.*

***הוסיפו*** *לקוד שלכם את השורה הבאה:*

plt.contour(coord\_x, coord\_y, V\_xy, np.sort([-1 , 0 , 1]), cmap='hot')

***ושנו*** *את הקוד כך שיצייר לפחות 9 קווים שווי פוטנציאל בערכים שונים (לא חובה שיהיו במרווחים אחידים, והם לא חייבים לכלול את -1,0,1). שימו לב ש-contour צריכה שה-levels יופיעו בסדר עולה, אך הפונקציה sort מבטיחה זאת עבורנו.*

***הריצו*** *את הקוד* ***ורשמו*** *בהערה מה הפרמטר cmap קובע.*

*כעת נצייר את הפוטנציאל שבין שתי האלקטרודות (על ציר X):*

1. ***הכינו*** *וקטור מתאים של x,* ***חשבו*** *עבורו את הפוטנציאל V\_x וציירו בגרף חדש את העקום בעזרת plt.plot(x,V\_x,'.', label=”calculated potential”)*

***מה למדנו?***

*סימלצנו פוטנציאל חשמלי (יצרנו עקום תיאורטי) והצגנו אותו בשלושה סוגי גרפים: (plot, pcolormesh, contour).*

## תרגיל - קבל לוחות

**רקע תאורטי**

קיבול הוא היחס בין המטען החשמלי למתח של גוף מוליך:

הקיבול של קבל לוחות עם לוחות עגולים בקוטר D ומרווח d בין הלוחות הוא:

*כאשר הוא המקדם הדיאלקטרי של ריק. כשנותנים לקבל C להיפרק דרך נגדים בעלי התנגדות כוללת של , המתח שעל הקבל דועך בצורה אקסופננציאלית:*

*כש- הוא קבוע הדעיכה. אם נבצע ln לשני האגפים נקבל קשר לינארי:*

*הזרם שזורם בקבל זורם גם בנגד R ולכן שווה למתח הנגד חלקי ההתנגדות:*

*מכיוון שהמטען הוא אינטגרל של הזרם, מקבלים שהשינוי במתח הקבל הוא:*

**המשימה:** עיבוד נתוני הניסוי

1. **פתחו** cell חדש בקוד בשם capacitor
2. נתון: . **חשבו** את הקיבול C\_theoretical

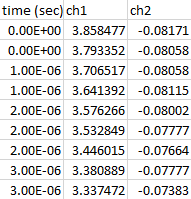
eps0 = scipy.constants.epsilon\_0 # F/m

D = 18e-2 # m

d = 0.5e-3 # m

1. נתון: . **חשבו** את tau\_theoretical

הקובץ capacitor.csv מכיל מדידות של הפריקה:



המתח על הנגד R הוא ch2 והמתח על הקבל הוא ch1-ch2.

1. **טענו** את הנתונים שבקובץ באופן הבא:

C\_data = pd.read\_csv('capacitor.csv')

**הסתכלו** ב-Variables Explorer איזה מידע מופיע ב-C\_data.

1. **שנו** את השם של עמודת הזמן ל- "t" ואת השם של “ch2” ל-"V\_R":

C\_data = C\_data.rename(columns = {"time (sec)":"t", "ch2":"V\_R"})

1. **הוסיפו** עמודה של ל-C\_data:

C\_data["V\_C"] = C\_data["ch1"] - C\_data["V\_R"]

1. **ציירו** את מתח הקבל כתלות בזמן. יש לציין label.

(**רמז:** למדנו לצייר גרף בתרגיל מיפוי פוטנציאל חשמלי – אתם יכולים להעתיק את הפקודות משם).

הערה: ניתן לשמור את הערכים שבטבלה במשתנים עם שם נוח יותר על ידי:

t = np.array(C\_data['t'].values)

V\_C = np.array(C\_data['V\_C'].values)

אם בחרתם באפשרות זאת – שימו לב כיצד יש להתאים את הקוד בהמשך והקלות של השימוש =)

1. **כתבו** תשובה מלאה בהערה בקוד: מה זה curve fitting?
2. **בצעו** fit לגרף בעזרת הקוד הבא, המכיל כתיבה של פונקציה המתארת את המודל אליו אנחנו מבצעים את ההתאמה וקריאה לפונקציה curve\_fit (אותה כינינו cfit בפקודת ה-import בתחילת הקוד):

def V\_decay(t,tau,V0):

return V0\*np.exp(-t/tau)

p\_optimal, p\_covariance = cfit(V\_decay,C\_data['t'], C\_data["V\_C"])

1. **רשמו** בהערה לשורה שמבצעת cfit מהם הפרמטרים שהפונקציה צריכה למצוא.
2. **חשבו** מתוצאת ה-cfit את V0\_fit ואת tau\_fit.
3. **קראו** בתיעוד של cfit מה המשמעות של p\_covariance **וחשבו** את השגיאה ב-V0\_fit ו-tau\_fit.
4. **הוסיפו** לגרף הנתונים את עקום ה-fit, והוסיפו מקרא (legend) באופן הבא:

plt.plot(C\_data['t'], V\_decay(C\_data['t'],p\_optimal[0],p\_optimal[1]),

label=’fitted curve’)

plt.legend()

האם העקום של ה-fit מתאים למדידות?

1. קראו בתיעוד של cfit על הפרמטר p0 והשתמשו בו עם הערכים המתאימים. עדכנו את הקריאה לcfit עם הניחוש ההתחלתי והריצו גם את שאר החישובים לאחריה. האם כעת העקום מתאים למדידות?
2. **חשבו** את מדד עבור עקום ה-fit בעזרת r2\_score(y\_measured,y\_predicted\_by\_fit). האם הערך שהתקבל סביר לדעתכם?
3. על גרף נפרד, **ציירו** את לוג המתח כתלות בזמן. אמור להתקבל קשר לינארי. **והוסיפו** grid: plt.grid()
4. לפי הגרף, **בחרו** מקטע זמנים בו מתקיים הקשר הלינארי, **צרו** וקטור של אינדקסים של מקטע זה, **וציירו** את המקטע שבחרתם:

inds = (C\_data['t'] > t1) & (C\_data['t'] < t2)

plt.plot(C\_data['t'][inds], np.log(C\_data["V\_C"])[inds],'.', label=’data’)

כעת נבצע רגרסיה לינארית כדי לחלץ את V0 ואת tau.

1. **כתבו** תשובה מלאה בהערה בקוד: מהי רגרסיה לינארית Linear Regression?
2. **בצעו** רגרסיה לינארית:

reg = linregress(C\_data['t'][inds], np.log(C\_data["V\_C"])[inds])

print(reg)

1. **חשבו** מתוך reg.slope ו-reg.intercept את V0\_reg ואת tau\_reg

הערה: אם תרצו לחשב קשר לינארי ללא intercept (כלומר intercept=0 במדויק) יש להשתמש ב-cfit.

1. חשבו את השגיאות ב-V0\_reg ו-tau\_reg בעזרת reg.stderr ו-reg.intercept\_stderr (שימו לב - יש לחשב שגיאות נגררות).
2. קראו על rvalue שמוחזר על ידי הפונקציה linregress **וחשבו** את (כלומר, ה- coefficient of determination) של הרגרסיה.
3. **הוסיפו** את עקום הרגרסיה לגרף (חשבו, כיצד אפשר לעשות זאת?) והוסיפו גם מקרא.
4. **בצעו** אינטגרציה למתח על הנגד ושמרו את התוצאה ב-C\_data[“int\_V\_R”]. (השתמשו ב-scipy.integrate.cumtrapz(V\_R, x = t, initial = 0))
5. בגרף חדש, **שרטטו** את השינוי במתח הקבל () כתלות באינטגרל של מתח הנגד וציינו label. אמור להתקבל קשר לינארי. אם הוא לינארי רק במקטע – **הגדירו** את האינדקסים של מקטע זה ב-inds2.
6. **בצעו** התאמה מתאימה (שימו לב, צריך לבצע linregress או cfit?) **וחשבו** את הקיבול C\_meas.
7. **הוסיפו** לגרף את עקום הרגרסיה, מקרא, grid וכותרות לצירים:

plt.xlabel("integral of V\_R")

plt.ylabel("$\Delta V\_C$")

plt.legend()

plt.grid()

התרשמו ממידת ההתאמה של הערכים התיאורטיים לערכים שחישבתם מהנתונים. הם צריכים להיות לפחות מאותו סדר גודל.

**מה למדנו?**

למדנו כיצד לטעון נתונים מקובץ, לבצע עליהם חישובים, להוסיף עמודות נתונים, לחתוך מתוך הנתונים את המקטע הרצוי, לבצע אינטגרציה, לבצע fit ורגרסיה לינארית ולעצב את הגרפים.

(אם אתם לא חושבים שזה מה שלקחתם מהתרגיל –אנחנו מציעים בחום, קראו שוב את הקוד שיצרתם - הכול שם)

שימו לב! כעת אתם יודעים לבצע את כל הפעולות הנדרשות לעיבוד הנתונים – ראו את [הרשימה מעלה.](#_חלק_ד_–)

**התרגילים הבאים הם תרגול של פעולות אלו.**

תרגיל – חוק אוהם

**רקע תאורטי**

כאשר זורם זרם בנגד, ההספק המתבזבז עליו הוא

*והאנרגיה הזאת הופכת לאנרגית חום:*

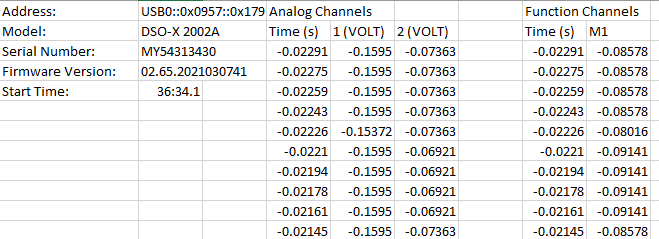
*טמפרטורת הנגד עולה, לפי קיבול החום שלו:*

*וכאשר הוא מתחמם, ההתנגדות של הנגד משתנה באופן לינארי:*

*כלומר, בסך הכול:*

*וזהו הקשר הלינארי אותו מוצאים בניסוי.*

*הקובץ ohm.csv מכיל את המדידות הבאות:*

**

כאשר המתח על הנגד המתחמם הוא המתח בערוץ 1 פחות המתח בערוץ 2, והזרם הוא המתח בערוץ 2 חלקי התנגדות R1 של 5.48 אוהם.

1. **צרו** section חדש בשם Ohm
2. **כתבו** פונקציה שמחשבת זרם: def I\_R(V2, R1)
3. **כתבו** פונקציה שמחשבת את המתח שעל הנגד: def V\_R(V1, V2)
4. **כתבו** פונקציה שמחשבת את התנגדות הנגד: def R\_t(V\_R, I\_R)
5. **כתבו** פונקציה שמחשבת את ההספק שמתבזבז בנגד: def P\_t(V\_R, I\_R)
6. **כתבו** פונקציה שמחשבת את האנרגיה שמחממת את הנגד: def Energy(P\_t, t)
7. **טענו** את הנתונים מהקובץ: R\_data=pd.read\_csv(“ohm.csv”, header=1)
8. **רשמו** בהערה מה המשמעות של הפרמטר header ולמה הוא נחוץ כאן.
9. **קראו** מה הפרמטר usecols של הפונקציה read\_csv מבצע **ותקנו** את השורה בקוד כך שרק העמודות של

Time (s) 1 (VOLT) 2 (VOLT)

יטענו לתוך המשתנה R\_data.

1. **שנו** את שם העמודה של הזמן ל-t ואת שמות הערוצים ל-V1, V2.
2. **השתמשו** בפונקציות שכתבתם וחשבו את התנגדות הנגד כתלות בזמן ואת האנרגיה כתלות בזמן.
3. **שרטטו** גרף של ההתנגדות כתלות באנרגיה.
4. **בצעו** רגרסיה לינארית (לקטע הלינארי) והוסיפו את העקום לגרף.
5. **חשבו** את ואת . **השוו** עם חברים לכיתה את התוצאה. האם קיבלתם ערכים דומים?

## תרגיל - השראות

בניסוי זה אתם תעבדו על סט של קבצי נתונים – **וזו חשיבות התרגיל!**

**רקע תאורטי**

כאשר יש שינוי בשטף המגנטי שבתוך סליל, מתפתח עליו מתח שנקרא כא"מ (כוח אלקטרו מניע) לפי הנוסחה הבאה:

במערכת הניסוי, מפילים מגנט דרך זוג סלילים ומודדים את המתחים שמתפתחים עליהם. מכיוון שהמתח תלוי במהירות המגנט, צריך לאפיין את התנועה שלו – כלומר, את המהירות ההתחלתית ואת התאוצה שלו.

לשם כך, נמדדו חמישה סטים של מתחים [Trace 0.csv, …., Trace 4.csv], כאשר הסליל העליון (ref) הוחזק במיקום קבוע והסליל התחתון (signal) הורחק ממנו ב-

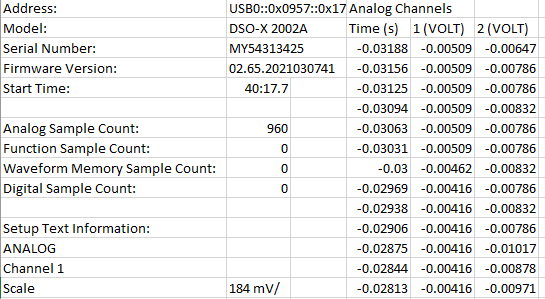
*סנטימטרים.*

*נגדיר את זמן המעבר בכל סליל בתור הזמן בו השטף הוא מקסימאלי. אם נקבע את להיות זמן המעבר בסליל הראשון, אז הזמן בו עובר המגנט את הסליל השני מקיים:*

*וניתן לרשום:*

1. ***פתחו*** *section חדש בשם inductance*

*קבצי המדידה נראים כך:*

**

*כאשר ערוץ 1 הוא המתח בסליל הראשון, הקבוע, וערוץ 2 הוא המתח בסליל הזז.*

1. ***הגדירו*** *בקוד את h בתור numpy array (np.array([…])) ושימו לב שהיחידות יהיו במטרים.*
2. ***טענו*** *את המדידות באמצעות לולאת for:*

*שימו לב לשם הקבצים.*

Ind\_data = []

for n in range(0,5):

df = pd.read\_csv('Trace %d.csv'%n, header = 1)

Ind\_data.append(df)

*משלב זה, אתם מוזמנים לשנות את שורות הקוד שבתוך לולאת ה-for כדי שיבצעו את החישובים הדרושים לכל קובץ נתונים.*

1. ***שנו*** *את השמות של וקטורי הנתונים: זמן ל-“t”, ערוץ 1 ל-“ref” וערוץ 2 ל-"signal"*
2. ***ציירו*** *בגרף את האותות signal ו-ref כתלות בזמן (את הנתונים מכל חמשת הקבצים באותו הגרף).*
3. ***כתבו*** *פונקציה לחישוב השטף מגנטי מתוך מתח. כלומר flux(voltage,time)*
4. ***ציירו*** *בגרף את השטף המגנטי המחושב מהמתחים ref ו-signal כתלות בזמן (כל הנתונים בגרף אחד).*
5. ***חשבו*** *את האינדקס בווקטור השטף בו השטף מקסימאלי (בערך מוחלט), ניתן להיעזר ב-np.argmax().*
6. ***הוסיפו*** *לגרף של השטף את הנקודות שמצאתם (plot(t[ind\_max]),flux[ind\_max], ‘ro’)). האם אלו הנקודות הנכונות?*
7. *לכל מדידה,* ***תקנו*** *את וקטור הזמן t כך שהזמן בו השטף מקסימאלי ב-ref מוגדר להיות t=0.*
8. *לכל מדידה,* ***חשבו*** *את זמן המעבר בסליל ה-signal (זמן בו השטף מקסימאלי, לאחר התיקון בסעיף הקודם ושמרו אותו בווקטור נתונים t\_coil.*
9. ***שרטטו*** *גרף לינארי מתוך h ו-t\_coil (לפי הנוסחה). מה צריך להיות ציר X? מה צריך להיות ציר Y?*
10. *נניח שגיאה של 0.1 ס"מ ב-h ושגיאה של 2 מילישניות בזמן t\_coil.* ***קראו*** *את התיעוד של הפונקציה matplotlib.pyplot.errorbar()* ***וצרו*** *גרף שמציג שגיאות אלו (יש לחשב שגיאות נגררות - ראו חומרי עזר באתר הקורס).*
11. ***חשבו*** *את המהירות ההתחלתית ואת התאוצה בעזרת רגרסיה לינארית. האם התקבלה התאוצה של נפילה חופשית?*
12. ***הוסיפו*** *את עקום הרגרסיה לגרף עם השגיאות.*
13. ***חשבו*** *את מדד R squared עבור הרגרסיה הלינארית.*

***מטלות סיכום:***

1. *נהוג להגדיר את הפונקציות (def) בתחילת הקוד, כדי למנוע קריאות שגויות. עבור כל section,* ***מקמו*** *את הפונקציות שכתבתם בתחילת הsection.*
2. ***רשמו*** *הערות לכל section – מה מבוצע בפקודות השונות ולאילו סעיפים הוא מתאים. ההערות צריכות להיות מספיק ברורות כך שתוכלו להבין אותן בעוד חודש.*
3. *לכל figure שיצרתם,* ***הוסיפו*** *את החלקים הדרושים לפי ההנחיות שבחומרי העזר, לא כולל CAPTION.* ***הוסיפו*** *גם grid. להגבלת הצירים לתחום מסוים יש להשתמש ב-xlim(), ylim().*
4. ***צרו*** *section* ***וכתבו*** *בו בהערות:*
   1. *רשימה של הפקודות שלדעתכם יהיו הכי שימושיות בניתוח נתונים במעבדה, מתוך הקוד שכתבתם עד כה.*
   2. *רשימה של הפקודות שמופיעות בקוד שלכם ושאתם לא בטוחים עד הסוף מה הן מבצעות.*
5. *בונוס:* ***הדפיסו*** *(print) את ערכי הקבועים e, mu\_0, epsilon\_0,m\_e בעזרת scipy.constants*

# חלק ה' – עיבוד נתונים מעבדת גלים

בחלק זה של התרגול, נלמד מספר דרכים חדשות לעיבוד נתונים והצגת מידע, שיהיו רלוונטיות למעבדה 3 – גלים. עיקרי התרגול הם:

1. יצירת והדפסת טבלה בעזרת משתנה Dictionary
2. יצירת גרף פולרי (שרטוט פונקציה של זווית ומרחק במקום X ו-Y)
3. עיבוד תמונה
4. מציאת שגיאות עם cfit

כדי לעבד את הנתונים, נשתמש בmodules הבאים:

import numpy as np # math functions

import scipy # scientific functions

import matplotlib.pyplot as plt # for plotting figures and setting their properties

import pandas as pd # handling data structures (loaded from files)

from scipy.stats import linregress # contains linregress (for linear regression)

from scipy.optimize import curve\_fit as cfit # non-linear curve fitting

from sklearn.metrics import r2\_score # import function that calculates R^2 score

from prettytable import PrettyTable # display data in visually table format

import scipy.ndimage as ndimage # import Multidimensional Image processing package

from scipy.signal import find\_peaks # for find local peaks in a signal

## תרגיל – ספקטרוסקופיה

בניסוי זה עליכם למדוד מספר קווי פיזור של הליום דרך סריג, ולתעד אותם לטבלה. לצורך כך נשתמש במשתנה בשם Dictionary

**רקע תאורטי**

כאשר אנו מקרינים אור ממנורת הליום דרך סריג, אורכי הגל השונים הנפלטים מהמנורה יתפזרו בזוויות שונות מהסריג, וכך נוכל למדוד את הספקטרום שלו. על מנת לחשב את אורך הגל של כל אחד מהפסים שנמדוד, נשתמש בנוסחה הבאה:

*כאשר הוא קבוע הסריג, n הוא סדר הפיזור, ו- היא הזווית המינימלית בה נמדד הקו. מכיוון שאנו רואים אך ורק את קווי הפיזור מסדר ראשון, נציב , ונמדוד את הזוית בה אנו רואים את הקו ביחס לסדר ה-0, ונתעד אותן בטבלה, לפי צבע, מספר, וזווית.*

*קבוע הסריג*

המשימה: הכנסת זוויות הפיזור למילון, חישוב אורכי הגל עבור כל קו פיזור, והצגת הנתונים בטבלה

**חשוב:** על מנת להתקין את ה-module עבור פונקציית pretty\_table, יש לגשת לקונסולה של אנקונדה:

Winkey->search for: anaconda prompt

ושם לרשום pip install prettytable

1. צרו קובץ חדש בשם spectroscopy.py
2. יבאו מקובץ ה-CSV בשם “Spectro\_dsta.csv” את הנתונים לתוך ווקטורים
3. צרו ווקטור בשם wavelengths, הממיר כל זווית מהנתונים לאורך גל לפי הנוסחה
4. הגדירו משתנה חדש, בשם Lines כמשתנה מילון: Lines = {}
5. הכניסו אליו את זוויות הפיזור באופן הבא
6. שימו לב כי בכל שורה של המילון, הערך השמאלי נקרא “key” והערך השמאלי נקרא “value”, ולדרך בה אנו מכניסים ערך של מספר לתוך מחרוזת בעזרת %d. בכל איטרציה של הלולאה, המילון מתעדכן בעזרת פונקציית update.

for n in range(0,len(angle)):

key=f'Line %d'%(n+1)

value=[color[n],angle[n],wavelength[n]]

**Lines.update**({key:value})

1. כעת נארגן אותם בטבלה באופן הבא: הגדירו משתנה חדש בשם table באופן הבא:

table = **PrettyTable**([“Grating Angle [deg]”,”Grating Wavelength [nm]”,”colour”])

1. הוסיפו לטבלה את הערכים שקיבלתם בלולאת For בעזרת הפקודה

**table.add\_row**([angle,wavelength,colour])

1. צרו ווקטור של ערכי זוויות הפיזור המתקבלות מהמנסרה. שימו לב, כי ישנם פסי פיזור אשר לא נצפו במנסרה! לכן על ערכים אלו להיות ריקים, דוגמא לווקטור עם ערכים ריקים:

A = [1,2,’ ‘,4,5,6,’ ‘,8]

1. הוסיפו עמודה עם כותרת “Prism Angle [deg]” וערכי הזוויות המתקבלות מפיזור המנסרה בעזרת הפקודה:  
   שימו לב כי כל זווית פיזור שהכנסתם לטבלה מתאימה לשורה המתאימה לה לפי קובץ ה-data.

**table.add\_column**(“Prism Angle [deg]”,data\_vector)

1. הדפיסו את הטבלה בעזרת print(table)

## תרגיל – מדידת קיטוב של גלי מיקרו

בניסוי זה אנו רוצים להראות כי הקיטוב של גלי מיקרו ביציאה מ"גלבו" הוא לינארי או מעגלי. נעשה זאת על ידי מדידת העוצמה של הגל כפונקציה של הזווית, ונציג את המידע בגרף פולרי. עבור קיטוב לינארי, נרצה לראות שינוי בעוצמה עם שינוי הזווית, ואילו עבור קיטוב מעגלי, נרצה לראות שהעוצמה נשארת קבועה.

**רקע תאורטי**

*גלבו, או מוליך גל, הוא מכשיר הבנוי ממוליך, אשר מוביל את הגלים דרכו.* גלבו מעביר גלים מקצה אחד לשני כך שהגלים לא מאבדים מעוצמתם. צורתו יכולה להשתנות אך היא תהיה קשורה לאורך הגל.

Shape

Description automatically generated

המשדר במערכת זו משדר גל מיקרו בקיטוב לינארי. על ידי סיבוב המשדר ב-45 מעלות לכיוון הגלבו, נקבל שני גלים, אחד בכיוון X ואחד בכיוון Y, בפאזה זהה. עקב תכונות הגלבו, הגל המקוטב בכיוון X יצבור פאזה בהתאם למרחק בין לוחות הגלבו. אם נקבע את המרחק בין הלוחות בצורה בה הפאזה הנצברת היא כפולה של , נקבל כי הקיטוב ביציאה מהגלבו הוא לינארי. לעומת זאת, אם נקבע את המרחק כך שהפאזה הנצברת היא , נקבל הפרש פאזה בין הגלים בכיוון X ו-Y ונקבל קיטוב מעגלי ביציאה.

1. צרו קובץ חדש בשם Microwave.py, ויבאו לווקטורים את הנתונים מקובץ ה-CSV בשם “microwave\_waveguide.csv”
2. צרו משתנים theta,Intensity עבור הזווית והעוצמה, והכניסו אליהם את הנתונים.
3. בעזרת matplotlib.pyplot, שרטטו גרפים פולריים עבור שני המקרים בעזרת הפונקציה: **plt.polar**(theta,Intensity). שימו לב כי הזווית נמדדת ברדיאנים.
4. קבעו עבור כל אחד מהגרפים האם מדובר בקיטוב לינארי או מעגלי

## תרגיל – האפקט האקוסטו-אופטי

בניסוי זה, נתרגל שימוש בעיבוד תמונה על מנת לחשב את אורכי הגל ומהירות הקול בנוזל לפי האפקט האקוסטו אופטי.

**רקע תאורטי**

*בניסוי זה מקרינים לייזר מונוכרומטי על תא מלא בנוזל. בנוזל זה מחוללים גלי קול בעזרת גביש פיזו-אלקטרי בתדר ידוע. משום שמהירות האור גדולה בהרבה ממהירות הקול בנוזל, האור הפוגע בנוזל יראה אותו כסריג עקיפה.*

*כאשר נצלם את האור המפוזר מיד לאחר תא הפיזור, בתחום השדה הקרוב, אנו נראה את המבנה הסריגי של הגלים בתא, ומהתמונה נוכל לחלץ את אורך הגל בנוזל, ולחשב את מהירות הקול בו.*

*לעומת זאת, אם נצלם את האור המפוזר רחוק מאוד מהתא, בתחום השדה הרחוק, אנו נראה נקודות התואמות את פיזור האור מסריג עקיפה. על ידי מדידת המרחקים בין הנקודות, נוכל לחשב את קבוע הסריג, ולקבל את מהירות הקול בנוזל.*

*שתי שיטות אלו משלימות אחת את השניה, שכן הן נותנות שתי דרכים למדוד את אותו הגודל. תכונה מעניינת של התמונות, השדה הקרוב והשדה הרחוק, היא שהן התמרת פורייה אחת של השניה.*

*עבור השדה הקרוב, מכיוון שהמרחק בין נקודות מינימום הוא אורך הגל, נקבל את הקשר הבא:*

*כאשר הוא אורך הגל הנמדד, ו-f הוא תדר המחולל.*

*עבור השדה הרחוק, נקבל את הקשר הבא:*

*כאשר הוא מוקד העדשה בה השתמשנו בניסוי, הוא המרחק בין הנקודות בתמונה, הוא אורך הגל של הלייזר (632.8 nm), ו-f הוא תדר המחולל.*

*עבור שתי התמונות, תדר המחולל הינו f = 2079 MHz, תדר הלייזר הינו ויחס ההמרה מפי'סלים למיקרונים הינו .*

1. פתחו קובץ חדש בשם Acousto\_optic.py

2. שמרו את התמונות באותה תקייה בה שמור קובץ העבודה שלכם

3. טענו את התמונה של השדה הקרוב בשם “NearField\_2079MHz.tif” למשתנה בעזרת הפונקצייה

image=**plt.imread**(“NearField\_2079MHz.tif”)

4. המשתנה מכיל מטריצה תלת מימדית, כאשר המימד הראשון הוא ערך ה-Y של הפיקסל, המימד השני הוא ערך ה-X שלו, והמימד השלישי מכיל את ערכי ה-RGB של התמונה בערכים של 0-255.

5. הציגו את התמונה עם פקודת imshow   
וצפו בתמונה בחלונית ה-plots.

**plt.imshow**(image)

6. סובבו את התמונה כך שהתמונה החדשה תהיה מקבילה לציר ה-X.

rotated\_image = **ndimage.rotate**(image,deg,reshape=True)

הציגו את התמונה שוב בעזרת plt.imshow כדי לראות את השינוי שעשיתם. בחרו בזווית מתאימה כך שקווי הסריג יהיו מאונכים לציר ה-X ככל הניתן

7. בחרו איזור בו ניתן לראות בבירור את המבנה הסריגי. צרו משתנה X בעזרת np.linspace באורך של 400 פיקסלים.

8. צרו משתנה Y עבור עוצמת האור בכל פיקסל

y = rotated\_image[y\_0,x\_0:x\_0+len(x),0]

בצורה זו, אנו לוקחים חתך של התמונה המקביל לציר ה-X, באורך של מספר הפיקסלים שקבענו קודם לכן, בצבע אחד (R במקרה הזה). בחרו את הערכים של x\_0,y\_0 כך שתקבלו איזור בו רואים את המבנה הסריגי של הנוזל

9. שרטטו בגרף את המשתנים שיצרתם. שימו לב כי אתם רואים את העלייה והירידה בעוצמה בגרף.

10. חשבו את המרחק בין מקסימות/מינימות עוקבות מהגרף שקיבלתם בעזרת find\_peaks, ושרטטו את הגרף יחד עם המקסימות שמצאתם בצורה הבאה (יכול להיות שפרמטר distance לא יתאים בדיוק ל:

peaks,\_=**find\_peaks**(y,distance=40)

plt.plot(y)

plt.plot(peaks,y[peaks],'^')

10. חזרו על שלבים 3-10 עבור תמונת השדה הרחוק בשם” FarField\_2079MHz.tif”. שימו לב להגדיר פרמטר distance מתאים על מנת למצוא את המקסימות הרלוונטיות בלבד.

11. חשבו את המרחק בין הנקודות, ובעזרת הנוסחה חשבו את מהירות הקול בנוזל. שימו לב כי אתם ממירים בין פיקסלים למטרים! השוו בין התוצאות שקיבלתם.

12. על מנת להראות שהתמונות הינן התמרת פורייה אחת של השניה, השתמשו בפונקציות **np.FFT** ו-**np.iFFT** על ווקטורי העוצמה שיצרתם עבור שתי התמונות. בדקו כי אתם מקבלים גרפים דומים לאחר ההתמרה.

## תרגיל – שני זמזמים בצינור

בתרגיל זה נתאים עקום בעזרת curve fitting עבור פונקציה לא פולינומית, נגדיר את הגבולות עבור הפרמטרים, ונקבע נקודת התחלה עבורם.

**רקע תאורטי**

כאשר שני זמזמים, המוזנים מאותו מקור, מונחים אחד מול השני בצינור, נקבל בצינור התאבכות משני המקורות. אם נמדוד את הגל בנקודה X בצינור, נקבל את תרומת שני המקורות:

*כאשר a היא משרעת הגלים, x\_1,x\_2 הם מרחקי המיקרופון מהזמזמים, f הוא תדר הגלים, אורך הגל, ו- הפרש פאזה כללי בין הזמזמים.*

*לאחר שימוש בזהויות טריגונומטריות, נראה כי נקבל פתרון של גל עומד, עם משרעת תלויה במיקום שנתונה לפי:*

*כאשר L הוא המרחק בין הזמזמים.*

1. צרו קובץ חדש בשם Amp\_fit.py
2. טענו את הנתונים לווקטורים מתאימים עבור המרחק מזמזמם אחד מתוך “2\_buzzers\_data.csv”, ועבור העוצמה הנמדדת
3. הגדירו פונקציה def f(x,a,lam,phi) אשר תואמת את פונקציית האמפליטודה ברקע התיאורטי. שימו לב כי על המשתנה הבלתי תלוי X להיות ראשון.
4. השתמשו במתודה:   
     
     
   על מנת לבצע התאמה לנקודות לפי הפונקציה שהגדרתם. לתוך p0 הכניסו ערכים התחלתיים לביצוע ההתאמה, וב-bounds הכניסו זוגות ערכים לפי סדר המשתנים עבור גבולות ההתאמה (לדוגמה, שימו לב מה יכולים להיות גבולות האמפליטודה, גבולות פאזה הם בין מינוס פאי לפאי, וכו').

params , cov = **cfit**(f, x\_data, y\_data, p0, bounds)

1. חשבו את ערך ה-R2 של ההתאמה באופן הבא:

y\_pred = f(x\_data, \*params)

Rsq = **r2\_score**(y\_data ,y\_pred)

1. שרטטו על גרף אחד את נקודות המדידה ואת קו הרגרסיה שקיבלתם
2. חשבו את ערכי השגיאה עבור הפרמטרים שקיבלתם מהרגרסיה על ידי לקיחת השורש של איברי האלכסון במטריצת cov.

*מזל טוב! הגעתם לסוף הסדנה!*

*כעת יש להגיש את הקוד שכתבתם במקום המתאים באתר.*

*שימו לב: קוד זה ישמש אתכם בעיבוד הנתונים במעבדה, אבל גם בניסויים האחרים הקוד הדרוש הוא דומה. אחרי שעבדתם כל כך קשה, תצטרכו רק להעתיק ולהדביק את החלקים הרלוונטיים.*

*טיפ למתקדמים: אפשר לקרוא את ההנחיות לביצוע לפני מפגשי המעבדה ולהכין קוד מותאם מראש לכל מפגש 😉 – חוסך זמן במעבדה ומקל להבין את הנתונים.*

*בהצלחה לכם!*