בס"ד תשפ"ב , מאי 2022

תרגיל תכנות שני- Python

רקע

מטרת התרגיל היא מימוש העקרונות המתמטיים שלמדנו עד כה בקורס לבעיות אמיתיות בעולם התקשורת.

תאריך הגשה אלקטרונית: 23.6.

הוראות הגשה כלליות:

- העבודה תוגש באתר MOODEL כתיקיה מכווצת בתיבת הגשה. **עליכם להוסיף לתיקיית התרגיל המכווצת שהורדתם קובץ pdf הכולל הסברים אנליטיים וגרפים, לעדכן את קובץ קוד המעטפת לקובץ הפתרון שלכם, ולצרף את קבצי הקוד שהכנתם. עליכם לוודא כי הקוד רץ בשלמותו.**
- העבודה תוגש בזוגות בלבד, כאשר כל זוג יגיש עותק אחד בלבד.

עבודה עם פייתון:

מומלץ להתקין סביבת עבודה עליה תעבדו.

ניתן להתקין anaconda בקישור הבא:

https://www.anaconda.com/products/individual/download-success

לאחר מכן ניתן להוריד אפליקציות שונות בסביבה עליהן תוכלו לערוך את הקוד.

מומלץ לפתוח סביבה משלכם עליה תעבדו.

הספריות שנשתמש בהן בתרגיל:

. sounddevice, matplotlib, soundfile,numpy

יש להתקין אותם לסביבה על מנת שהקוד ירוץ.

אפליקציות מומלצות:

PyCharm

Spyder

VS Code

מבוא

בתרגיל זה נעבוד עם אותות בזמן בדיד, נחשב עבורם את מקדמי הפורייה וטור הפורייה.
ראינו בקורס כי טורי פורייה הם עבור אותות מחזוריים בלבד, בתרגיל זה ניקח אות לא מחזורי, נחלק אותו למקטעים, ולכל מקטע בתורו נתייחס כמחזור יחיד של אות מחזורי.
לצורך המטלה עליכם להכין את הפונקציות הבאות –

```
def FourierCoeffGen(signal):
def  DiscreteFourierSeries(FourierCoeff):
```

1. הפונקציה FourierCoeffGen מקבלת כקלט אות כניסה באורך מחזור אחד. הפונקציה מחשבת את מקדמי הפורייה a_k של האות.
   שימו לב – האות שהכנסנו לא בהכרח מחזורי. אנחנו חותכים מקטע מהאות הרצוי, ומבצעים לו "המשכה מחזורית".
   בפונקציה עליכם למצוא מהו זמן המחזור-N ולחשב את סדרת המקדמים של האות a_k לכל ערכי k (זכרו כי אנחנו בזמן בדיד, כמה מקדמי פורייה אמורים להיות?)
   עליכם להחזיר את מקדמי הפורייה של האות בלבד.

2. הפונקציה DiscreteFourierSeries מקבלת כקלט את מקדמי הפורייה של האות- a_k ומחשבת את טור הפורייה שלו. שוב, עליכם להפיק מתוך סדרת המקדמים מחזור יחיד באורך N.

חלק א

חלק זה משמש לבדיקת הפונקציות אותן כתבתם בלבד, לא צריך להגיש אותו כחלק מהקוד, אלא להציג ולהסביר את התוצאות שקיבלתם בקובץ הPDF.

הכניסו לפונקציה ()FourierCoeffGen את האותות הבאים:

$$x_1[n] = \cos\left(\frac{2\pi * n}{C}\right)$$

$$x_2[n] = \begin{cases} 1 & for\ |n| < 5N_1 \\ 0 & else \end{cases}$$

מחזורי במחזור $N = 20N_1$

כאשר C זוהי הספרה האחרונה בתעודת הזהות של אחד השותפים, N_1 הספרה האחרונה בתעודת הזהות של השותף השני (אם אחד מהערכים הוא 0, אתם יכולים לבחור כל מספר שלם שתרצו בתחום 1-9).

שימו לב שעליכם להכניס לפונקציה מחזור בודד בלבד!

חשבו אנליטית את מקדמי הפורייה השוו את התוצאות האנליטיות לתוצאות בפייתון.
מתוך מקדמי הפורייה מצאו את טור הפורייה, בדקו האם התוצאות שקיבלתם הגיוניות.
צרפו גרפים רלוונטים והסברים מתאימים.

כעת, השתמשו בסדרת המקדמים שקיבלתם על מנת לבנות מחדש את האות בזמן, באמצעות הפונקציה () DiscreteFourierSeries

השוו בין האות המקורי לאות המשוחזר.

חלק ב

כעת עליכם לעבור לקוד הראשי speech_slow_down.
בתרגיל זה נביר אלגוריתם המתמודד עם בעיה מוכרת בעולם התקשורת.

בערוצי תקשורת המידע מתחלק למקטעים (Packets) בגודל קבוע וכך האות נשלח בערוץ מקטע אחר מקטע (לעיתים עם חפיפה בין המקטעים)
לדוגמא, שיחת זום, כדי שנוכל לנהל שיחה בזמן אמת- אות הדיבור מתחלק למקטעים וכך עובר למשתמשים האחרים בשיחה.

תופעה מוכרת בעולם התקשורת היא איבוד מקטעים או עיכוב שלהן. במקרה כזה נצפה שלא נוכל להעביר מידע בין משתמשים בצורה אמינה ויהיו מעין "חורים" ברצף המידע.
כדי לשמור על שיחה רציפה בין המשתמשים הוצע אלגוריתם אותו נממש בתרגיל.

כאשר יש עיכוב בערוץ, כדי ליצור המשכיות ולא ליצור הפסקים בשיחה הוצעה השיטה הבאה:
נניח שפקטה מספר N מתעכבת למשך M-1 זמני שידור, במקרה כזה נשדר את פקטה $(N-1)$ למשך M-1 פעמים בנוסף לפעם המקורית (סה"כ M פעמים). לאחר הגעת הפקטה נחזור לשדר את פקטה N. במקרה כזה אות הדיבור יישמע "מרוח" אבל לא תיהיה הפסקה בדיבור כלל.

אנחנו בתרגיל נממש את האלגוריתם עבור M קבוע. כלומר כל הפקטות יועכבו במספר קבוע של זמני שידור.
בנוסף, כדי שאות הדיבור יישמע טבעי יותר, יש לדאוג לכך שהפאזה של האות תיהיה רציפה.
כלומר בכל פריים נדאג שתיהיה קפיצת פאזה זהה.

תחילה נצטרך לחלץ את הפאזה מתוך מקדמי הפורייה:

$$\measuredangle a_k = \frac{a_k}{|a_k|}$$

נחשב את הפרש הפאזה בין הפאזה של המקטע הנוכחי לפאזה של המקטע הקודם:

$$phase_{diff} = \frac{\measuredangle a_k}{phase_{pre}}$$

לאחר מכן נדאג להוסיף לכל מקטע חדש את אותו הפרש הפאזה מהמקטע הקודם:
כלומר:

$$current_{phase} = last_{phas} \cdot phase_{diff}$$

$$b_k = |a_k| \cdot current_{phase}$$

את החישוב הזה נעשה בלולאה M פעמים כדי לדאוג להשהיה הנכונה.
יש לזכור לשמור את הפאזה האחרונה לחישוב הבא בלולאה.

הערה:

מסיבות נוספות של שיפור האלגוריתם מקובל לבחור את המקטעים עם חפיפה של 75% ולכפול אותם בחלון hamming לפני חילוץ המקדמים ושוב אחרי בניית הטור. חלק זה מצורף לקוד ולא תצטרכו לממש אותו.

חלק ג

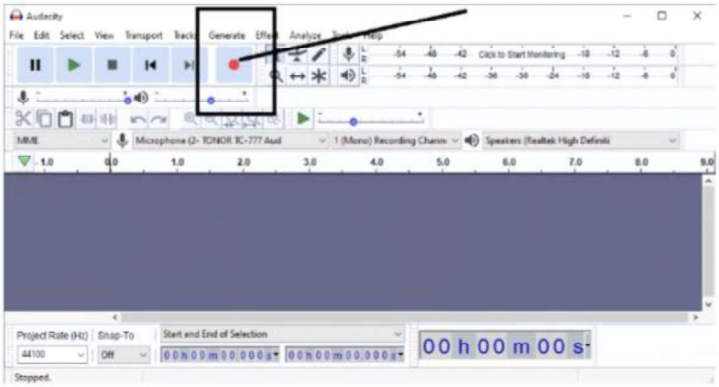
הקליטו את עצמכם, המירו את ההקלטה לפורמט ושמעו את האות במוצא עבור ערכי M שונים.

הערה:

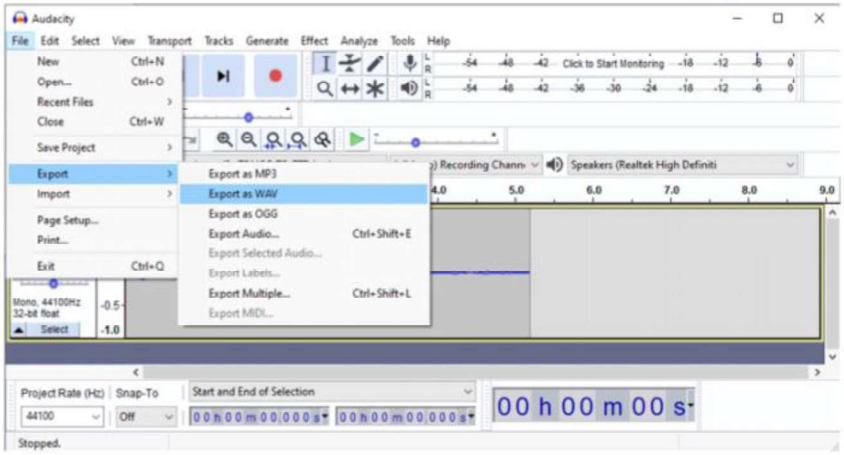
האת ההקלטה יש לטעון ב- python ב**פורמט wav**.

בחרו בדרך נוחה לכם על מנת להקליט/להמיר. להלן כמה אפשרויות:

1. להקליט בטלפון/במחשב (voice recorder בווינדוס) ולהמיר את הקובץ באינטרנט באתרים המבצעים המרה בין פורמטים של קבצי אודיו.
2. להקליט במחשב בעזרת **תוכנת Audacity** – תוכנה פשוטה ושימושית מאוד!
   a. הורדת התוכנה: https://www.audacityteam.org/
   b. הקליטו את עצמכן/ם. לחצו על הכפתור המסומן:



   c. ייצאו את הקובץ בפורמט wav:



בהצלחה! 😊

פייתון

יום רביעי 15 יוני 2022 20:02

316411578 yaniv hajaj
322573452 ori glam

- העבודה תוגש באתר MOODEL כתיקיה מכווצת בתיבת הגשה. **עליכם להוסיף לתיקיית התרגיל המכווצת שהורדתם קובץ pdf הכולל הסברים אנליטיים וגרפים, לעדכן את קובץ קוד המעטפת לקובץ הפתרון שלכם, ולצרף את קבצי הקוד שהכנתם. <u>עליכם לוודא כי הקוד רץ בשלמותו.</u>**
- העבודה תוגש בזוגות בלבד, כאשר כל זוג יגיש עותק אחד בלבד.

חשבו אנליטית את מקדמי הפורייה השוו את התוצאות האנליטיות לתוצאות בפייתון.
מתוך מקדמי הפורייה מצאו את טור הפורייה, בדקו האם התוצאות שקיבלתם הגיוניות.
צרפו גרפים רלוונטים והסברים מתאימים.

$$x_1[n] = \cos\left(\frac{2\pi * n}{C}\right)$$

$cos(\omega_0 n)$ $\omega_0 = \frac{2\pi m}{N}$	$\pi[\delta(\omega - \omega_0) + \delta(\omega + \omega_0)]$	$a_k = \begin{cases} \frac{1}{2} & k = \pm(m + l \cdot N) \\ & l \in \mathbb{N}_0 \\ 0 & \text{אחרת} \end{cases}$

טור פורייה בדיד:

נוסחת הסינטזה:

$$x[n] = \sum_{k=\langle N \rangle} a_k e^{jk\omega_0 n}$$

נוסחת האנליזה:

$$a_k = \frac{1}{N} \sum_{n=\langle N \rangle} x[n] \cdot e^{-jk\omega_0 n}$$

```
############################################################################ cos signal
C = 8  # yaniv hajaj last ID digit
N = C  # one period equal C
pie = math.pi  # use pi from library math
e = math.e
x_1 = []  # array of x1 points
for n in range(N):
    x_1.append(math.cos((2 * pie * n) / C))  # N element of x1 samples W=2pi/c -> N=2pi/w -> N=c
plt.plot(x_1)  # X_1 on a graph
plt.title("first signal cos(2pi*n/c)")  # X_1 title
plt.show()
cos_coeff = FourierCoeffGen(x_1)
reConstructedSignal = DiscreteFourierSeries(cos_coeff)
plt.plot(reConstructedSignal, 'g')
plt.title("reconstructed signal in time - cos")
plt.show()
############################################################################
```

גרף של האות הראשון שבנינו COS

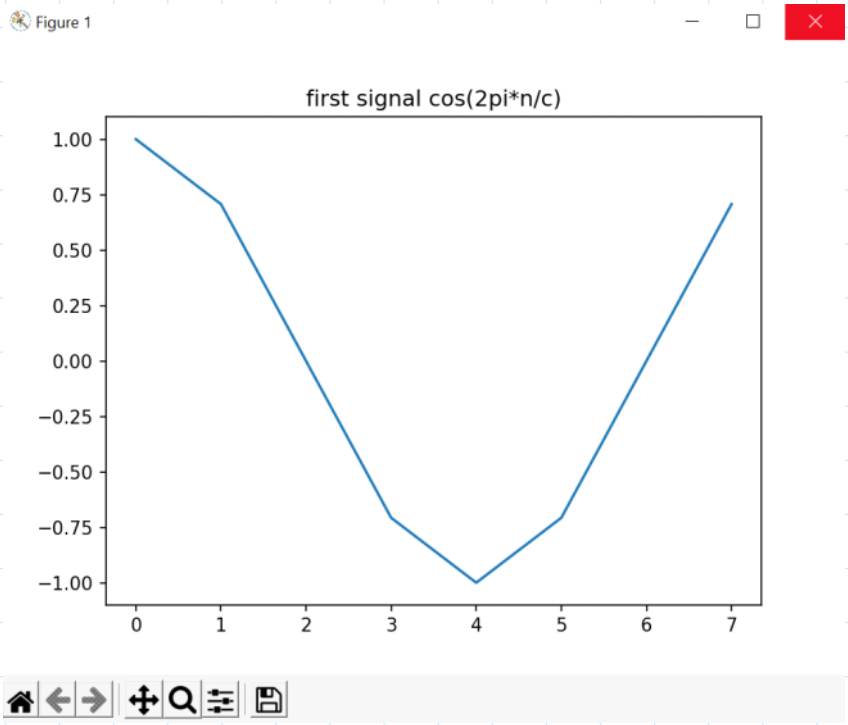
$$x_1[n] = \cos\left(\frac{2\pi * n}{C}\right)$$

לקחתי $c=8$ הסברה האותרונה בתצוגת הבהוב
k יניב חצץ

$$x_1[n] = \cos\left(\frac{2\pi n}{8}\right) = \frac{1}{2}\left[e^{j\frac{2\pi n}{8}} + e^{-j\frac{2\pi n}{8}}\right] \Rightarrow a_1, a_{-1} = \frac{1}{2}$$

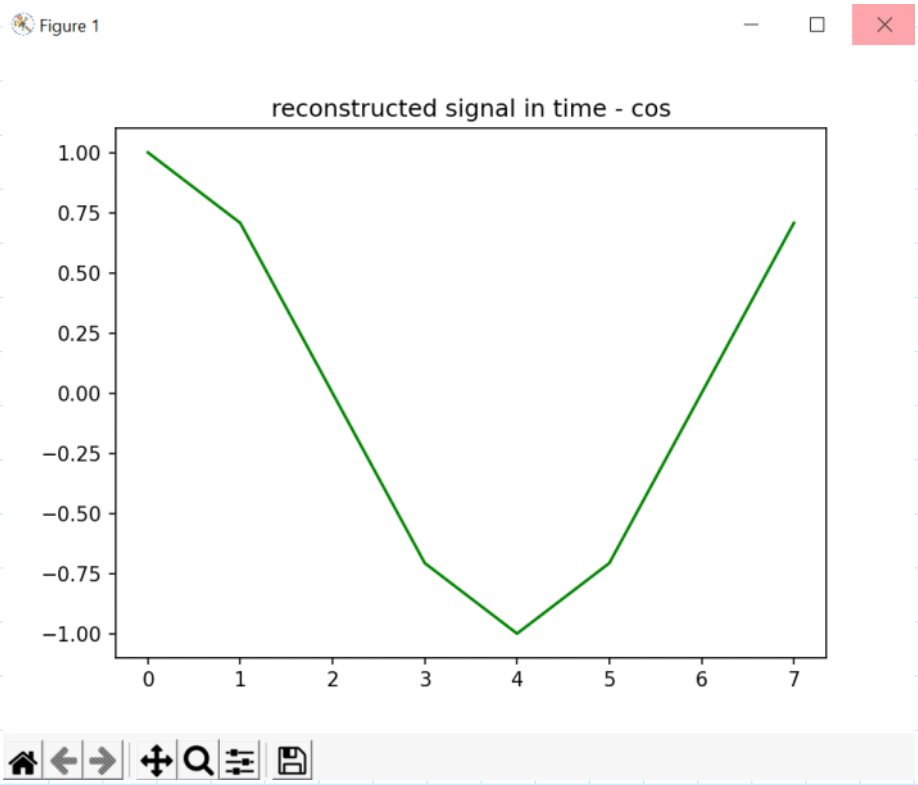
$\hat{w}$ $\hat{w}$

$a_1 = a_{-1} = a_8 = a_{-8} = \frac{1}{2}$ והמקדמים מתאפסים לכן



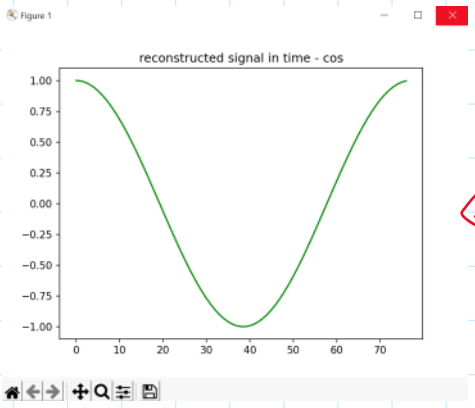
הכנסנו את האות לפונקציית מקדמי הפורייה וחילצנו את מקדמי הפורייה.

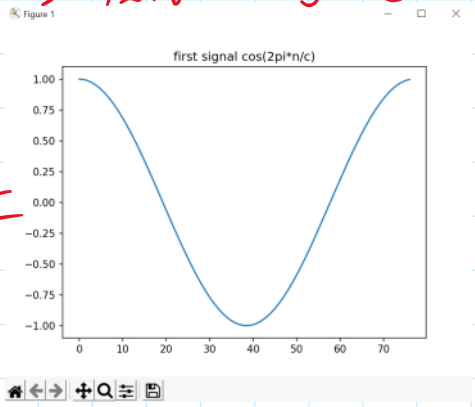
```python
def FourierCoeffGen(signal):
    # TODO: Implement the FourierCoeffGen function.
    # This function compute signal's Fourier Coefficients.
    # Fourier Coefficients for discrete signals are periodic a_k(k)=a_k(k+N)
    # so we will compute only N
    w_0 = (2 * pie) / N  # 2pi divided by the period
    i = complex(0, 1)  # put 0+i inside i
    FourierCoeff = []  # array of coeef

    for k in range(N):  # get N FourierCoeff, each one from sigma
        a_k = 0
        for n in range(N):  # sigma of DFT
            a_k = a_k + signal[n] * np.exp(-i * k * w_0 * n)  # sum of signal*exponent
        a_k = a_k / N  # divided by N (period) (normalization)
        FourierCoeff.append(a_k)  # put new coef to FourierCoeff Array

    return FourierCoeff
```



ניתן להוות כי המחצור היה אוכן ל
עבור ערכים 0-7 נקל נקודות מהווות $x[n]$
ובנקודה $x=0$ הוות שווה לנקודה $x=7$

עבור $c=77$ רואים באופן ברור
שהבנייה עובדת





אות מספר 2

$$x_2[n] = \begin{cases} 1 & for\ |n| < 5N_1 \\ 0 & else \end{cases}$$

בתעודת בהיר

לקחנו $N_1=2$ מספר אחרונה של אורך

$$x_2[n] = \begin{cases} 1 & for\ |n|<10 \\ 0 & else \end{cases}$$

$$\boxed{a_k = \begin{cases} \dfrac{\sin\left(\frac{2\pi k}{N}\left(N_1+\frac{1}{2}\right)\right)}{N\sin\left(\frac{2\pi k}{N}\right)} \end{cases}}$$

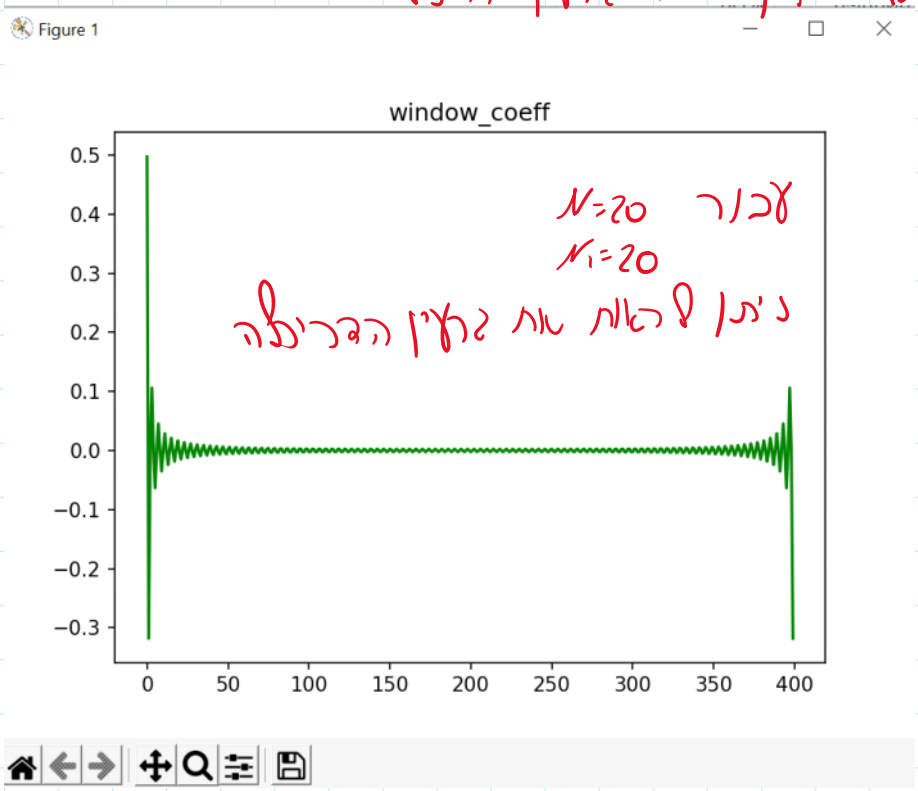
נשמאות שהגל

נהיה שהוא סוג של גרעין פירלה

נראה שהגוף של חלון דריכלה
לקחתי עבור $N_1=20$ כדי שהיה קל לראות את הגרף
עם הדגימות בגודל $N=20$

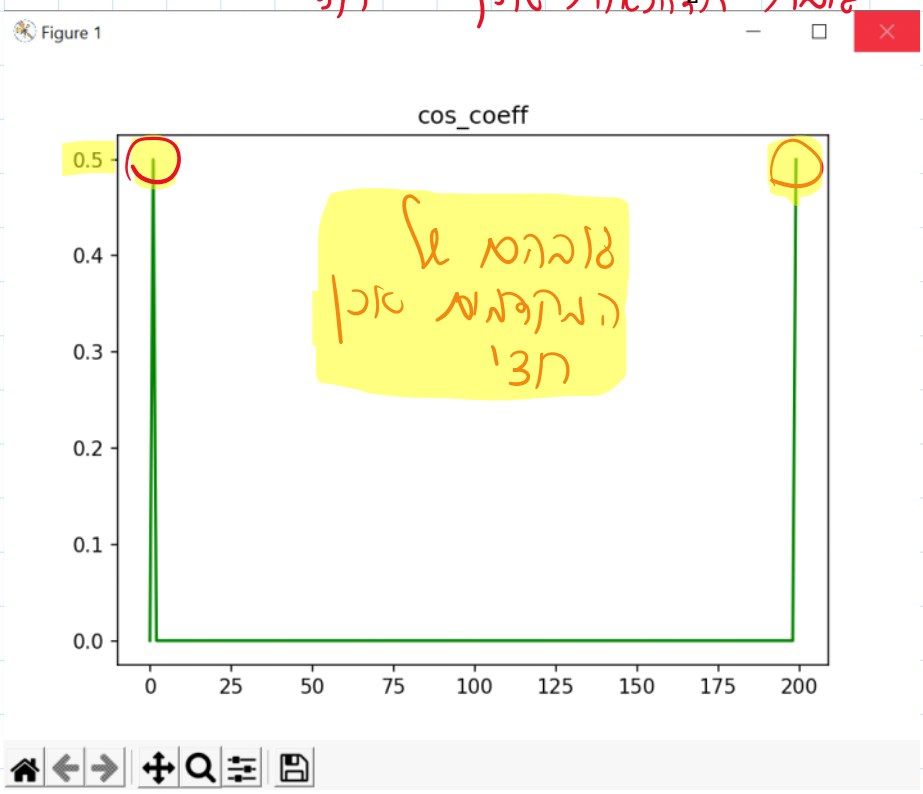
חלון דריכלה ← $a_k = \dfrac{\sin\frac{2\pi k}{20}\left(20+\frac{1}{2}\right)}{20\sin\left(\frac{2\pi k}{20}\right)}$

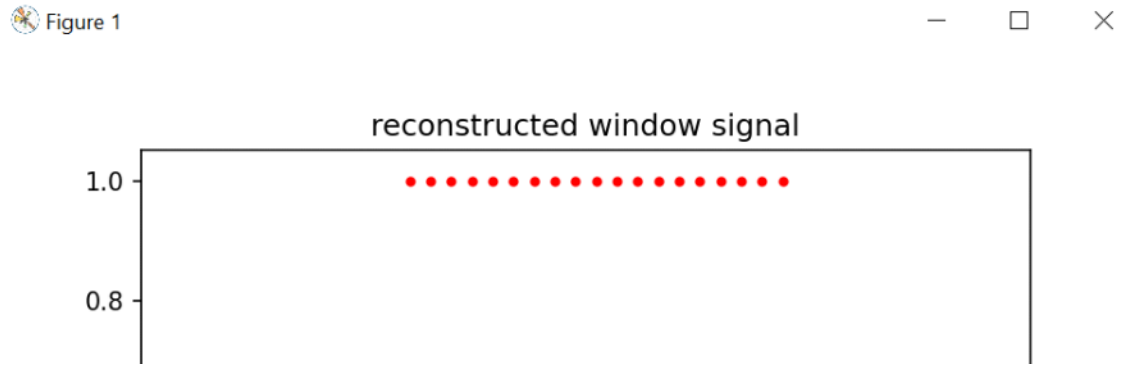
ואז הגינו כי מקדמי סדרייה
של חלון הם חלון דריכלה

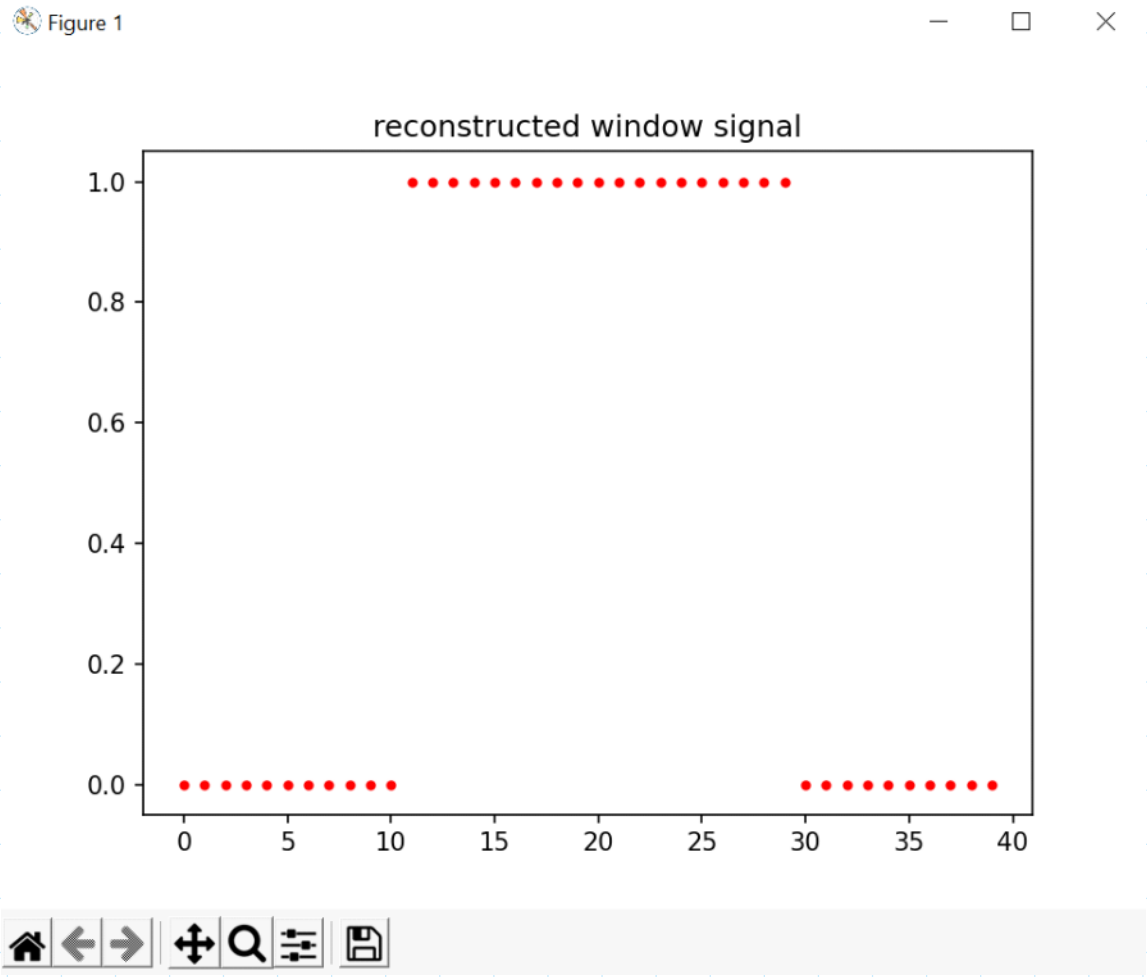


```python
##########################################################################
N_1 = 2  # ori glam last ID digit
N = 20 * N_1
x_2 = []
for n in range(int(-N / 2), int(N / 2)):  # sample from -N/2 to N/2
    if abs(n) < 5 * N_1:  # |n|<5*N_1 put 1
        x_2.append(1)
    else:  # |n|>5*N_1 put 0
        x_2.append(0)
plt.plot(x_2, 'r.')
plt.title("window signal")
plt.show()
coe = FourierCoeffGen(x_2)
reconstructedSignal = DiscreteFourierSeries(coe)
plt.plot(reconstructedSignal, 'r.')
plt.title("reconstructed window signal")
plt.show()
##########################################################################
```

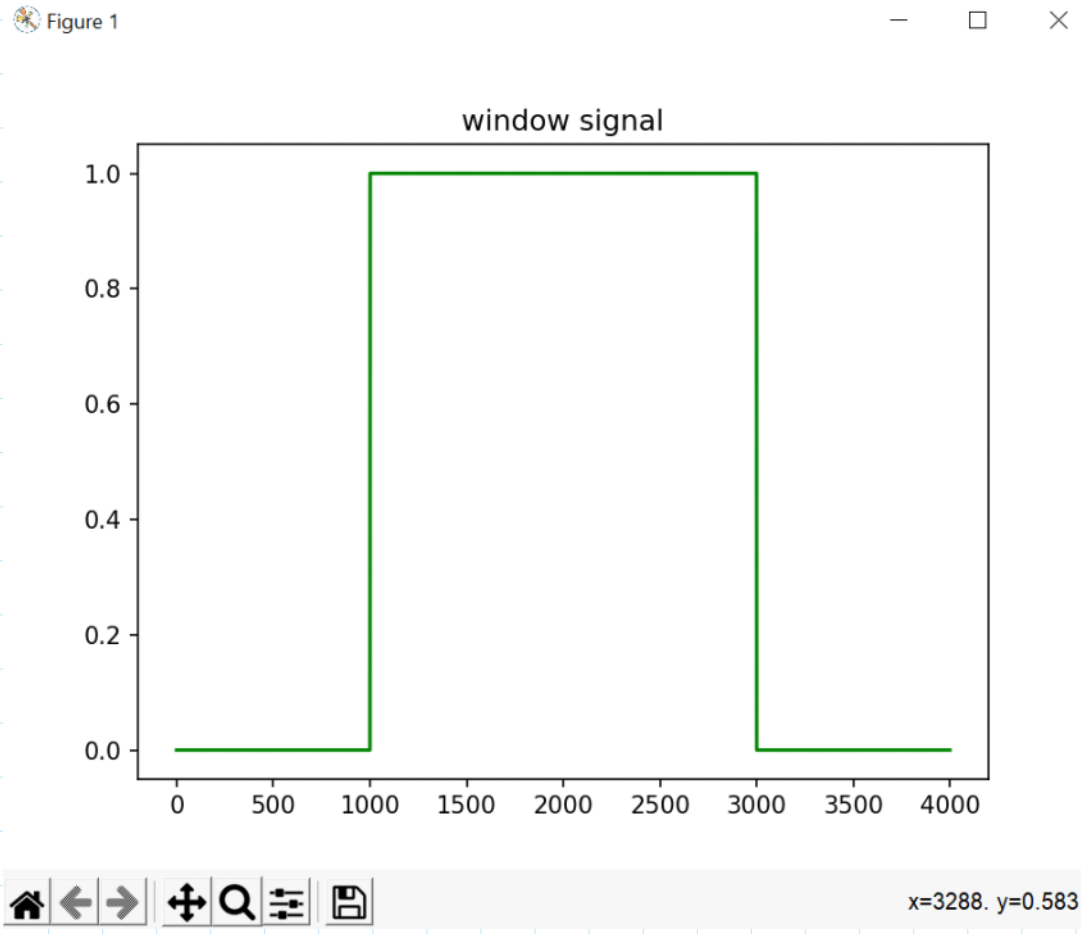
הנבסנו סדרה של 200=C כדי שנראה בבירור את הפלטאוות
גובה הפלטאות אכן חצי







לתהליך שלה $N_1 = 200$ בגדול ויקבלנו מעין סינקציית כלון כבדרוש



בחלק ג'

* ביצענו את המבוקש בחלק ג' עבור הקלדות שונות עם ערכי חת שונים ומצאנו כי ככל שחת עלה כך שמע הבאות נעשה איטי יותר.

כמו כן הקלטנו את עצמנו והעלינו להגשה שם ההקלטה שהוספנו בקבוצת YaniVslow