תרגיל באינטגרציה נומרית ומציאת שורשים

מולקולה דו-אטומית, כגון O_2, מורכבת משני גרעינים הנקשרים על ידי האלקטרונים המקיפים אותם. מכיוון שהגרעינים כבדים הרבה יותר מהאלקטרונים, ניתן להניח שהאלקטרונים נעים מהר מספיק כך שהם מסתדרים מיידית בהתאם לשינוי המיקום של הגרעינים (קירוב בורן-אופנהיימר). תנועת הגרעינים נשלטת על ידי פוטנציאל אותו ניתן לרשום כפוטנציאל לנרד-ג'ונס:

$$V(r) = 4V_0\left[\left(\frac{a}{r}\right)^{12} - \left(\frac{a}{r}\right)^{6}\right]$$

הערך המזערי של הפוטנציאל מקבל ב-$r_{\min} = 2^{\frac{1}{6}}a$ והוא שווה ל-$-V_0$.

את תיאור התנועה של הגרעינים ניתן לפשט יותר על ידי הפרדה בין סיבוב (רוטציה) לבין שינויי מרחק מהירים (וויברציה). תנועת הוויברציה מתוארת על ידי המצבים הקשורים $\psi_n(r)$ המתקבלים מפתרון משוואת שרדינגר חד-ממדית:

$$\left[-\frac{\hbar^2}{2m}\frac{d^2}{dr^2} + V(r)\right]\psi_n(r) = E_n\psi_n$$

כש-m זו המסה המצומצמת של שני הגרעינים ו-E_n היא האנרגיה של המצב ה-n-י.

מכיוון שהמסה של הגרעינים היא גדולה למדי, ניתן לפשט עוד יותר את הבעיה ולתאר את הפתרון על ידי חישוב תנועה קלאסי בפוטנציאל V והפעלת "כלל קוונטיזציה" כדי לקבוע את האנרגיה. כללים אלו, שנקבעו על ידי בוהר וסומרפלד ווילסון היוו את הבסיס לתאוריית הקוונטים "הישנה". תנועה קלאסית קשורה בפוטנציאל $V(r)$ קיימת עבור אנרגיות $-V_0 < E < 0$, כאשר המרחק בין הגרעינים מתנדנד מחזורית בין רדיוס מזערי r_{in} ורדיוס מרבי r_{out} עבורם $V(r) = E$. במהלך התנודות האנרגיה מוחלפת מאנרגיה קינטית של תנועה יחסית לאנרגיה פוטנציאלית כך שסך האנרגיה קבועה: $E = \frac{p^2}{2m} + V(r)$, כש-p הוא התנע היחסי של הגרעינים. מכיוון שכך, ניתן לבטא את התנועה באנרגיה נתונה כמקיימת מסלול סגור במרחב הפאזה (p, r) כך ש-:

$$p(r) = \pm\{2m[E - V(r)]\}^{\frac{1}{2}}$$

בתנועה קלאסית מתאפשר כל ערך של E בין 0 ל-$-V_0$. כללי הקוונטיזציה מאפשרים רק ערכים המקימים את הקשר הבא:

$$S(E_n) = \oint \frac{1}{\hbar}\, p(r)dr = 2\left(\frac{2m}{\hbar^2}\right)^{\frac{1}{2}}\int_{r_{in}}^{r_{out}}[E_n - V(r)]^{\frac{1}{2}}dr = \left(n + \frac{1}{2}\right)2\pi$$

כלומר, השטח של המסלול במרחב הפאזה ביחידות של $\hbar$ מקבל ערכים חצי שלמים של 2π.

ניתן לרשום את הגדלים באופן חסר יחידות:

$$\epsilon = \frac{E}{V_0} \ , \qquad x = \frac{r}{a} \ , \qquad \gamma = \left(\frac{2ma^2V_0}{\hbar^2}\right)^{\frac{1}{2}}$$

ואז מקבלים:

$$(*) \quad s(\epsilon_n) \equiv \frac{1}{2}S(\epsilon V_0) = \gamma \int_{x_{in}}^{x_{out}} [\epsilon_n - v(x)]^{\frac{1}{2}} dx = \left(n + \frac{1}{2}\right)\pi$$

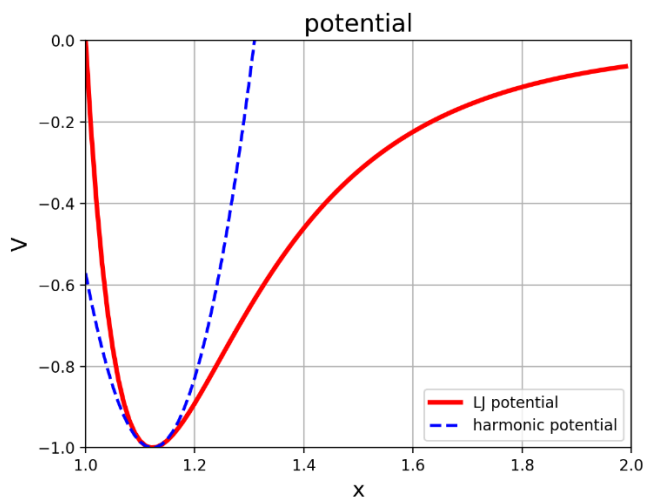
כש-:

$$v(x) = 4\left(\frac{1}{x^{12}} - \frac{1}{x^6}\right)$$

ככל שהמסה גדולה יותר, או $\hbar$ קטן יותר, הבעיה מתקרבת לגבול הקלאסי ו-γ גדל. עבור מולקולת O_2 מתקבל $\gamma = 150$.

יש לכתוב תוכנית מחשב המחשבת את האנרגיות (חסרות היחידות) ϵ_n עבור מולקולת O_2.

ניתן לבדוק את התוכנית על ידי החלפת הפוטנציאל לפרבולי והשוואה לפתרון של תנועה הרמונית. כלומר יש לפתח את הפוטנציאל סביב נקודת המינימום ולבצע את החישוב עם פוטנציאל פרבולי ולהשוות את האנרגיות המתקבלות.



בשלב ראשון עדיף לעשות בדיקה בסיסית יותר ולבחון את האנרגיות המתקבלות מפתרון $(*)$ עבור פוטנציאל הרמוני פשוט: $v(x) = x^2 \ , \ \gamma = 1$ בהשוואה לפתרון האנליטי שהוא: $\epsilon_n = 2n + 1 \ ,$ $n = 0,1,2,3, \ldots$

הסבר מפורט על מימוש התרגיל וחלוקה למטלות משנה מופיעים בעמודים הבאים.

לצורך הפתרון יש לכתוב שתי פונקציות נומריות:

1. אינטגרציה נומרית של פונקציה חד-ממדית: לצורך חישוב האינטגרל ב-(*) כלומר לחישוב : $s(\epsilon_n) = \gamma \int_{x_{in}}^{x_{out}} [\epsilon_n - v(r)]^{\frac{1}{2}} dx$, כאשר את גבולות האינטגרציה x_{out} , x_{in} ניתן לחשב מביטוי אנליטי המתקבל מהצבת $v(x)$ ודרישה ש-: $\epsilon_n = v(x_{in}) = v(x_{out})$.
2. מציאת שורש של פונקציה חד-ממדית: למציאת ϵ_n המקיים: $s(\epsilon_n) = \left(n + \frac{1}{2}\right)\pi$

<u>פונקציית האינטגרל:</u>

להגדיר פונקציה המחזירה קירוב נומרי ל-$I = \int_a^b f(x)dx$

f_integral(f,a,b,eps,global_type,integral_type)

global_type הוא אלגוריתם חלוקת הקטע המבוקש, כשיש ליישם שתי שיטות:

א. "fix_segments" - מחלקים את הקטע [a,b] ל- int(1/eps) גדלים שווים.

ב. "recursive" - מחלקים את הקטע [a,b] באופן רקורסיבי לשני קטעים תוך בקרת הדיוק כפי שתואר בקורס. הפרמטר eps מייצג את הדיוק האבסולוטי בחישוב האינטגרל (כדאי להגדיר ערך מזערי של 2 ומרבי של 50 לעומק הרקורסיה).

integral_type הוא סוג אלגוריתם האינטגרציה במקטע, כשיש ליישם שתי שיטות:

א. "simpson" שיטת סימפסון בה נקודות האינטגרציה בקטע [-1,1] הן $-1,0,+1$ והמשקלות הן $\frac{1}{3},\frac{4}{3},\frac{1}{3}$ בהתאמה.

ב. "gauss" שיטת גאוס-לג'נדר מסדר 3 בה נקודות האינטגרציה בקטע [-1,1] הן $-\sqrt{0.6}, 0, +\sqrt{0.6}$ והמשקלות הן $\frac{5}{9},\frac{8}{9},\frac{5}{9}$ בהתאמה.

<u>פונקציית השורש:</u>

להגדיר פונקציה: x_root(f,a,b,epsx,epsf,type) המחזירה שורש נומרי עבורו abs(f(x))<epsf בקטע [a,b]. type הוא האלגוריתם המבוקש כשיש ליישם שתי שיטות: "bisection" שיטת החציה (או "ציד אריות"), ו-"secant" בה משולבת גם שיטת החציה (כפי שהוסבר בקורס). בשתי השיטות תחום החיפוש מצטמצם במהלך החיפוש ויש להחזיר ערך גם אם גודל המרווח קטן מ-epsx. יש לוודא שסימן הפונקציה f הוא הפוך בשתי הקצוות בתחילה ולעצור את הביצוע עם הודעת שגיאה אם התנאי לא מתקיים.

מטלות:

1. לממש פונקציה לאינטגרציה כפי שתואר לעיל ולבדוק את האפשרויות השונות על ידי השוואה של פתרון נומרי ופתרון אנליטי לאינטגרציה של פונקציה אנליטית לבחירתכם. יש לבחון את מספר הקריאות לפונקציה (עליה עושים אינטגרציה) ולהגיש את תוצאות הבדיקות עם הסבר.
2. לממש פונקציה למציאת שורש כפי שתואר לעיל. גם כאן יש לבדוק את האפשרויות השונות על ידי השוואה של פתרון נומרי ופתרון אנליטי למציאת שורש של פונקציה אנליטית לבחירתכם. יש לבחון את מספר הקריאות לפונקציה (שאת השורש שלה מחפשים) ולהגיש את תוצאות הבדיקות עם הסבר.
3. לחשב נומרית את הפתרון של $(*)$ עבור פוטנציאל הרמוני פשוט: $\gamma = 1$, $v(x) = x^2$ בהשוואה לפתרון האנליטי שהוצג לעיל עבור $n = 0,1,2,3,4$. יש להשלים את הקוד הבא (היכן שמופיע "...") ולהגיש את התוצאות עם הסבר:

```
import numpy as np
def v_sq(x):
    global mone_v
    mone_v += 1  # counts how many times v was calculated for efficiency check
    return x*x

gamma = 1

def s(energy) :
    global mone_s
    mone_s += 1  # counts how many times s was calculated for efficiency check
    x_1 = …
    x_2 = …
    ff = lambda x …
    return gamma * f_integral(ff,x_1,x_2,1e-6,’recursive’,’gauss’)

for n in range (5):
    f = …
    mone_v, mone_s = 0, 0
    e_n = x_root(f,0,1000,1e-3,1e-3,’secant’)
    print('%3d %20.16f %6d %4d %12.4e %12.4e' %(n,e_n,mone_v,mone_s,f(e_n),e_n/(2*n+1)-1))
```

4. לחשב את הפתרון של (*) עבור פוטנציאל הרמוני שהוא קרוב של פוטנציאל לנרד-ג'ונס כפי שהוצג בציור בעמוד 2. יש להשלים את הקוד הבא (היכן שמופיע "...") ולהגיש את התוצאות עם הסבר:

```
import numpy as np
def v_hr(x):
    global mone_v
    mone_v += 1  # counts how many times v was calculated for efficiency check
    return …

gamma = 150

def s(energy) :
    global mone_s
    mone_s += 1  # counts how many times s was calculated for efficiency check
    …
    return gamma * f_integral(ff,x_1,x_2,1e-6,'recursive','gauss')

en_hr = np.zeros(14)
m_v = np.zeros(14)
m_s = np.zeros(14)
f_n = np.zeros(14)
for n in range (14):
    …
    mone_v, mone_s = 0, 0
    en_hr[n] = x_root(f,-.999,-1e-7,1e-3,1e-3,'secant')
    m_v[n], m_s[n], f_n = mone_v, mone_s, f(en_hr[n])
print('%20.16f' %(2*(-1-en_hr[0])))
for n in range (14):
    print('%3d %20.16f %6d %4d %12.4e %20.16f' %(n,en_hr[n],m_v[n],m_s[n],f_n[n],en_hr[n]-en_hr[n+1]))
```

5. לחשב את הפתרון של (*) עבור פוטנציאל לנרד-ג'ונס (חישוב en_lj[n]). החישוב דומה לזה שבסעיף 4 כאשר יש לחשב עד (כולל) $n = 38$. יש להדפיס את התוצאות בדומה להדפסה בסעיף 4 ולהסביר את התוצאות.

6. להשוות את התוצאות של סעיף 4 וסעיף 5 על ידי הוראות הציור הבאות ולהגיש את הציור והסבר.

```
from matplotlib import pyplot as plt
plt.figure(figsize=(6,5))
n1 = np.arange(14)
plt.plot(n1[:],en_hr[0:14],'o',label='harmonic-approx')
n2 = np.arange(39)
plt.plot(n2[:],en_lj[0:39],'x',label='lenard-Jones')
plt.grid()
plt.legend()
plt.show()
```

7. לבצע בדיקה של השפעת פרמטר הדיוק על חישוב ϵ_{20} על ידי השלמת הקוד הבא והגשת התוצאות עם הסבר

```
import numpy as np

def v_lj(x):
    global mone_v
    mone_v += 1
    return …

gamma = 150

def s(energy) :
    global mone_s
    global ep
    mone_s += 1
    …
    return gamma * f_integral(ff,x_1,x_2,1e4*ep,’recursive’,’gauss’)

n = 20
ep = 1e-3
print('bisection - recursive - gauss ')
print(' %3s %20s %12s %6s %4s %12s' % ('n','     en_p     ',' ep  ','it_v','it_s','   err   '))
for ip in range(10):
    …
    mone_v, mone_s = 0, 0
    en_p = root_1d_lh(f,-.999,-1e-7,ep,ep,’secant’)
    print(' %3d %20.16f %12.4e %6d %4d %12.4e' % (n,en_p,ep,mone_v,mone_s,f(en_p)))
    ep = ep/10
```

8. חזרה על סעיף 5 תוך שימוש בפונקציות נומריות מתוך scipy. שימוש בפונקציות אלו מודגם בקוד הבא:

```
import numpy as np
from scipy import optimize
from scipy import integrate

f = lambda x : np.exp(x)
int_f = lambda x : np.exp(x)
a,b  = 0.1, 20.
num_int = integrate.quad(f,a,b)[0]
print(num_int,int_f(b)-int_f(a))

f = lambda x : np.sin(x) - np.sin(0.7)
print(optimize.brentq(f,0,np.pi/2))
```