סדנה 3 - חישוב תכונות חומרים באמצעות סימולציית דינמיקה מולקולרית

21 בדצמבר 2020

1 מבוא

למה אנחנו פה בכלל?

חומר היא תופעה מורכבת. החומר עצמו מורכב מחלקיקים קטנים שלכל אחד מהם תכונות **מיקרוסקופיות** כמו מסת חלקיק, קבועי משיכה\דחייה ביניהם וכו' שאותם יחסית קל לנו למדוד באמצעות ניסויים. לעומת זאת החומר כולו מפגין תכונות **מאקרוסקופיות** כמו לחץ, טמפרטורה וכו'. ההבדל הוא שבעוד התכונות המיקרוסקופיות הן קבועות יחסית (מסת החלקיק לא משתנה), התכונות המיקרוסקופיות משתנות בהתאם לתנאי הסביבה. המשמעות היא שכל מדידה של תכונות מאקרוסקופיות היא "ניסוי" בפני עצמו שמלווה בזמן ומשאבים רבים.

השאלה הפיזיקלית שאיתה אנו מתמודדים היא האם מתוך התכונות המיקרוסקופיות של החומר אנו יכולים, באופן תיאורטי וחישובי לחלוטין, ללמוד על התכונות המאקרוסקופיות של החומר? בשביל זה אנחנו צריכים לבנות סימולציה שתדע לבצע את הקישור בין הגדלים השונים.

השאלה של תכונות חומרים מאד חשובה בתכנון של אמל"ח, ולכן ידע תיאורטי מלא של החומר נדרש בכדי לבצע אופטימיזציות וחישובים תיאורטיים לגבי טיב הפעולה של סוגי אמל"ח שונים. דמיינו לדוגמא כיצד ניתן לתכנן מטען חלול מבלי להכיר לעומק את תכונותיה של הנחושת? (למי שאינו מכיר - מטען המתבסס על סילון נחושת חם ומהיר לפריצת מיגון של טנקים ומשוריינים נוספים). בסדנה הנוכחית ננסה ללמוד טיפה על התכונות הפיזיקליות של הנחשות.

יותר חשוב מכך - הקונספט של דינמיקה מולקולרית אמנם תקף לחישוב חומרים, אך ניתן להכליל אותו לכל מערכת שבה אנו רוצים ללמוד על התכונות הגלובליות מתוך תכונות הפרטים הבודדים שבמערכת. אפשר לדוגמא לדמיין להקה של ציפורים שפועלים ביניהן כוחות דחייה ומשיכה, וללמוד מתוך הסימולציה על תכונות שונות של הלהקה כולה. יותר מ"קוד שמריץ חלקיקים" מה שאנו עושים פה הוא פיתוח אלגוריתמי ורעיוני של שיטה חישובית המאפשרת לנו ללמוד מהפרט אל הכלל.

2 שלב 1 - חישוב תכונות חומר בטמפרטורה $T = 0$

בגלל שסימולציה של דינמיקה מולקולרית היא גדולה ומורכבת נחלק אותה לשלבים. בכדי להוריד מכם את נטל תכנון הקוד ניתן לכם גם את החתימות של הפעולות השונות שנממש בכדי לייצר תהליך מימוש קל יותר.

בכדי לייצר טיפה סדר תעבדו ב־4 קבצים עם השמות - physics.py, dynamics.py, plot.py, main.py כך שכל קובץ יכלול את הפעולות הבאות:

- physics.py - יכלול את הפיזיקה של הסימולציה ואת הפעולות המתארות חישובים פיזיקליים\נומריים
- dynamics.py - יכלול את הפעולות הקשורות לדנימיקה של הסימולציה כולה (הלולאה הראשית, קריאה להדפסות וכו')
- plot.py - קובץ אשר כולל את פעולות העזר שמדפיסות דברים לגרפים\קבצים
- test.py - הקובץ הראשי שישמש אתכם להרצת הבדיקות השונות של הקוד שתעשו. כל פעולה בקובץ היא "בדיקה" של אחד הדברים בקוד.

1. תחילה פתחו את תרגיל 3 מהמודל ובצעו את שאלה 3 בו שירדה מהתרגיל. האצת הפעולה המופיעה שם היא קריטית להמשך החישובים שלכם. את הפעולות הללו שימו בקובץ physics.py.
2. **ויזואליזציה -** כתבו קוד שנעזר בפעולה LennardJonesPotential מתרגיל 3 ומשרטט את הגרף של הפוטנציאל, ודאו שהגרף תואם את מה שראינו מהשיעור. הוסיפו את הפעולה לקובץ plot.py. כמו כן הוסיפו בדיקה של הפעולה לקובץ test.py.
3. **בדיקת שפיות -** כתבו קוד שמוודא את הפעולה LennardJonesForce ע"י כך שהוא בוחן את הכוח בשני מקרים - מרחק גדול מאד בין החלקיקים (בחרו r_c גדול בהתאם), ומרחק r_m, כלומר בנקודת המינימום של הפוטנציאל. הוסיפו את הפעולה לקובץ test.py.
4. **שיפור יעילות -** נסו לשפר את היעילות של של הפעולות הבאות עד כמה שניתן, השתמשו בטכניקות שראינו בכיתה. סעיף ללא ניקוד, אך הוא ישפר את זמן הריצה שלוקח לכם עבור כל חישוב. באופן כללי - חישובים יכולים להיות איטיים, בניגוד לשיעורים קודמים יכול להיות שתצטרכו להריץ כ־10^{-5} חישובים לכל גרף ולא מאות.
5. כתבו עוד 3 פעולות נוספות והוסיפו אותן לתוך קובץ בשם physics.py. הפעולות הן:

   (א) הפעולה:

```
def system_energy(r_old, r, r_new, dt, L, rc):
        ...
```

היא הפעולה המחשבת את האנרגיה של המערכת. הפעולה מקבלת 3 "מערכי r" בשמות r_old, r ו־r_new המייצגים את המיקומים של חלקיקי המערכת בצעד הזמן הקודם, צעד הזמן הנוכחי וצעד הזמן העתידי. בנוסף הפעולה מקבלת גם את צעד הזמן dt, אורך צלע הקופסא L והקבוע r_c המייצג את ערך הסף לחישוב האנרגיה הפוטנציאלית של בין שני חלקיקים. למי מכם שאינו זוכר כל משתנה r הוא מערך דו־ממדי כך ש־ r[i,:] הוא מערך עם שני איברים אשר מייצג את מיקומי ה־x, y של החלקיק ה־i.

הפעולה מחשבת את האנרגיה הקינטית של המערכת (זכרו מהשיעורים איך הגדרנו את המהירות) ואת האנרגיה הפוטנציאלית של המערכת (ע"י סכימת האנרגיה הפוטנציאלית בין כל **זוגות** החלקיקים).

הפעולה תחזיר את האנרגיה הקינטית, האנרגיה הפוטנציאלית והאנרגיה הכוללת של המערכת.

(ב) הפעולה:

```
def verlet_step(r_old, r, dt, L, rc):
        ...
```

אשר שמות המשתנים בה זהים במהותם לשמות המשתנים בפעולה הקודמת. הפעולה מבצעת צעד Verlet כפי שהוגדר בשיעור ומחזירה את המיקום בצעד הזמן הבא r_new ואת האיבר הויריאלי virial כפי שהוגדר בסעיף 1 בפעולה של חישוב הכוחות.

(ג) הפעולה:

```
def pressure_virial(virial, Ek, L):
        ...
```

הפעולה מקבלת את ה־virial מהפעולה הקודמת, את האנרגיה הקינטית שחושבה קודם לכן ואת אורך הקופסה L ומחשבת את הלחץ של המערכת כפי שהוגדר בשיעור בעזרת הנוסחה הויריאלית (שימו לב שמדובר ב"לחץ רגעי" אותו נצטרך למצע על פני מספר נקודות זמן). שימו לב שאנו נותנים לכם את האנרגיה הקינטית ולא את הטמפרטורה. היעזרו בקשר $kT = \frac{1}{d} m \left\langle v_i^2 \right\rangle_\tau$ בכדי לקשור בין האנרגיה הקינטית של **כל** המערכת לבין הטמפרטורה שלה.

6. לאחר כתיבת הפעולות הללו בעצם השלמתם את "המנוע הפיזיקלי" של הקוד שלכם. כעת נשאר ל"הדביק" את הפיזיקה עם אתחול המשתנים הנדרשים, כתיבת ה"לולאה הראשית" של הקוד אשר תריץ את הפיזיקה בכל צעד מחדש והדפסת התוצאות הסופיות. לשם כך אנחנו נכתוב פעולה הבאה בקובץ dynamics.py:

```
def T0_config(dt, N, L, rc):
        ...
```

אשר מקבלת את צעד הזמן dt, כמות החלקיקים במערכת N, אורך המערכת L ומרחק הסף r_c. הפעולה הזו כוללת את הלוגיקה הבאה שבעזרת נוכל לחשב את לחץ החומר בטמפרטורה 0.

(א) אתחול חלקיקי החומר במיקומים רנדומליים בתוך התיבה.

(ב) ביצוע לולאה ראשית - הלולאה הראשית רצה כל עוד מתקיים תנאי התכנסות כלשהוא עליו נדון בהמשך. כל עוד אין התכנסות הלולאה תבצע צעד verlet בכדי לקבל את מיקומי החלקיקים בצעד הזמן הבא. לאחר מכן הפעולה "תחתוך" את המיקום החדש של החלקיקים לתוך הקופסא בעזרת הפעולה np.remainder. לאחר מכן הפעולה תעדכן את הקוד לצעד הבא ע"י ההשמה:

$$r_{old} \longleftarrow r_{new}$$

$$r \longleftarrow r_{new}$$

למה אנחנו מציבים את r_{new} גם בr_{old} וגם ב־r? אנחנו רוצים לחשב את הלחץ של המערכת ב־$T = 0$. ההשמה $r_{old} \leftarrow r_{new}$ בעצם "לוקחת" מכל חלקיק את כל האנרגיה הקינטית שלו (זכרו - את מהירות החלקיק מחשבים באמצעות r_{old}), ולכן מורידה את הטמפרטורה של כלל המערכת. בצורה הזו אנו דואגים שבכל צעד יש פחות אנרגיה קינטית מהאנרגיה הקודמת עד אשר נגיע ל־$T = 0$. תוכלו לחשוב על זה כאילו אתם מפילים כדור במדרון של הר, ובכל צעד דואגים להחזיק את הכדור כך שאין לו אנרגיה קינטית - כאשר הכדור יגיע לנקודה הנמוכה ביותר (ללא אנרגיה קינטית) הוא ייעצר בה.

(ג) לאחר סיום הלולאה הראשית - הפעולה תחזיר את r_{new}.

7. **בדיקת התכנסות** - חשבו על תנאי עצירה טוב עבור הסימולציה שיחליט מתי הלולאה בסעיף 2 תפסיק. חשבו שאתם רוצים לעצור את הסימולציה כאשר הטמפרטורה של המערכת קרובה ל־0, ולכן תצטרכו לחשוב על תנאי פיזיקלי המתאר את הקרבה הזו.

8. **בדיקת שפיות** - הריצו את הקוד עם 5 חלקיקים ועם $L = 2$, בשביל צעד הזמן בחרו $dt = 10^{-4}$ (אם אתם רואים בעיות בתוצאות נסו להקטין את צעד הזמן).
הוסיפו פעולה ל־plot.py אשר מקבלת "מערך r" כמו r_{new} ומשרטטת את מצב המערכת, כלומר גרף דו־ממדי עם צירי x, y שכל נקודה בגרף מציינת מיקום של חלקיק במרחב.
שרטטו גרף כזה עבור r_{new} המתקבל מהסימולציה שהרצתם. מהו המרחק שאתם מצפים לו בין זוג חלקיקים בטמפרטורה $T = 0$? ודאו שאכן מתקבל המרחק הזה. הוסיפו את הבדיקה לקובץ test.py.

9. ערכו את הפעולה שכתבתם בפעולה 5. הוסיפו לקוד 4 מערכים Energy, Pressure, Temperature, Counter.
עדכנו את הלולאה הראשית כך שפעם ב־1000 איטרציות תוסיף לכל אחד מהמערכים האלה את ערך האנרגיה, הלחץ, הטמפרטורה ומספר האיטרציה הנוכחי. היעזרו לשם כך בפעולות שרשמתם בסעיפים הקודמים. בנוסף עדכנו את הפעולה כך שתחזיר בנוסף ל־r_{new} את ארבעת המערכים הנ"ל.
תוכלו לחשוב על כל חישוב כזה כל 1000 איטרציות בתור "דגימה" של המערכת שבניתם. אפשר לחשוב על זה בתור ניסוי קטן שאתם עושים על המערכת ומודדים גדלים שונים בה. שימו לב לערוך את הבדיקות שכתבתם בהתאם.

10. **בדיקת שפיות** - שרטטו גרף של הטמפרטורה כתלות במספר האיטרציה - האם הוא מתנהג כמו שאתם מצפים שהוא יתנהג? הוסיפו את הפעולה ל־plot.py ואת פעולת הבדיקה ל־test.py.

11. **יחידות מצומצמות -** עד עכשיו עבדנו בחומר LJ כללי. היעזרו באינטרנט ומצאו את הקבועים הרלוונטיים עבור נחושת. היעזרו בעדכון שביצעתם בסעיף 8 ומצאו את הלחץ של נחושת בטמפרטורה 0. לשם כך עבדו עם 15 חלקיקים (אם תראו שהקוד שלכם רץ מהר מספיק תוכלו לבחור יותר חלקיקים) ושנו את L כך שצפיפות החלקיקים תשתנה בהתאם. שרטטו גרף של הלחץ כתלות בצפיפות (בטמפרטורה 0). הוסיפו

לכל נקודה בגרף גם את השגיאה שלה (היעזרו בשיעור מונטה־קרלו לראות איך מחשבים שגיאה של מערך של מספרים), הוסיפו את השגיאה על כל נקודה באמצעות errorbar על הנקודה. כרגיל הוסיפו את פעולת השרטוט ל־plot.py ואת פעולת הבדיקה ל־test.py.