Számítógépes szimulációk IV.: Molekuladinamika

Pál Balázs*

*Eötvös Loránd Tudományegyetem

2019. március 25.

Abstract

A *Számítógépes szimulációk* laboratórium negyedik alkalmával körüljártuk a termodinamika és a statisztikus fizika, azok által a fizikában képviselt szemléletmódját. Ez jelen esetben azt jelentette, hogy a szimulációkban nagyszámú, egymással kölcsönható részecske mikroszkopikus mozgását tanulmányoztuk, melynek segítségével a vizsgált rendszer egyes makroszkopikus tulajdonságait szerettünk volna feltérképezni. Ezek közé tartozott a rendszer egyensúlyi helyzetének vizsgálata, valamint az ilyen állapotban mérhető nyomás, a teljes energia, a kompresszibilitási faktor, valamint a hőkapacitás modellezése. Emellett megismerkedtünk a Verlet- és a Velocity–Verlet-algoritmusokkal, valamint azok korrekcióival, melyeket gyorsaságuk miatt előszeretettel alkalmaznak molekuladinamikai szimulációkban.

1. BEVEZETŐ

A labor negyedik alkalmával a molekuladinamika témakörével foglalkoztunk. Általánosítva olyan rendszereket vizsgáltunk, melyekben nagyszámú, egymással kölcsönható részecske található. A natív, eddig megismert integráló módszerekkel ezek megoldása a mai technológia számára még túl sok időt és elérhetlenül nagy számítási teljesítményt igényelne, így azok helyett másokat kell alkalmaznunk.
A most újonnan tanult módszereket Loup Verlet, francia fizikus javasolta elhíresült papírjában (Verlet, 1967[1]). Ezek - numerikus hibákat javító korrekciókkal ellátott verziói - akár tízszer gyorsabb sebességre képesek, kvázi ugyanolyan pontossággal, mint a Runge-Kutta függvények. Mivel egy nagy részecskeszámú rendszerben a legtöbb idő a részecskék közti erők kiszámítására fordítók, ezért a Verlet-féle algoritmusok ismerete és használata nagy segítséget nyújt minden, ilyen fajta szimulációban.

2. FELADATOK

A kitűzött feladatokon történő munka megkezdése előtt meg kellett ismerkednünk az előzetesen kiadott három darab programkóddal. Ezek mindegyike egyedül a Velocity–Verlet-algoritmust implementálta, és a szimulációk során a továbbiakban is kizárólag én ezt az egy iteratív algoritmust használtam. Az egyes kódok a Velocity–Verlet-algoritmus újabb és újabb korrekcióival bővítik az előtte levőt, folyamatosan pontosítva és gyorsítva a molekuladinamikai szimuláció modellét.
Fel kellett ismernünk, hogy kezdetben mindegyik szimuláció egy adatfile-t generál, melyben a minden egyes lépésben kiszámított pillanatnyi hőmérsékletek listája volt található. Ezek alapján azt kellett vizsgálunk, hogy azonos kezdőfeltételek mellett mennyi idő alatt relaxál a rendszer, annak egyensúlyi helyzetéhez.
Következő feladatunk a Verlet-féle szomszádsági listával és a távoli potenciálok levágásával operáló korrekciót megvalósító, valamint az enélkül integráló módszerek futásidejnek összehasonlítása volt. Az általam vizsgált karakterisztika a potenciálok levágási határának, valamint a szomszédsági lista frissítési intervallumának módosítására történő futásidőbeli változásokat foglalta magába. Mind a korrekciókkal ellátott, mind pedig az azok nélküli iteratív módszereket teljesen azonos paraméterekkel futtattam, az egyes futások között folyamatosan változtatva a levágási hossz és a frissítési időköz nagyságát. Kellően sok párosítás segítségével kellően jól fel tudtam térképezni az kérdéses különbségeket.
Utolsó előtti feladatként implementáltam a szimulált rendszer egyensúlyi pozíciójában mérhető nyomást, teljes energiát, kompresszibilitási faktort és hőkapacitást kiszámító függvényeket.
Végezetül a feladat a programkódok olyan átalakítása volt számunkra, ami egy keményfalú rendszer képes szimulálni az addigi határfeltétel nélküli, vagy periódikus módszerek mellett. Emellett opcionálisan az előző feladatban szereplő háromtest szimulációt ábrázolni képest programot kellett írnunk a molekuladinamika feladatai között megadott kód segítségével. Ezt az utóbbi már az előző feladat esetén megvalósítottam (lásd YouTube[2]), a keményfalú rendszerek dinamikáját pedig legelső lépésben implementáltam mindegyik különböző módszerre.
A fenti feladatokat is mind ezen zárt feltétel mellett vizsgáltam, többek között a nyomást is a falra kifejtett erőből számoltam.

3. ELMÉLETI ALAPOK

3.1. INTEGRÁLÓ MÓDSZEREK

Mind a sima Verlet-, mind pedig a Velocity–Verlet-algoritmusoknak megvannak a saját előnyei és hátrányai. Míg a Verlet-módszer majdnem olyan pontos, mint a negyedrendű Runge-Kutta integrálás (a hiba alig $\mathscr{O}\left(\tau^4\right)$ nagyságrendű, a Runge-Kutta $\mathscr{O}\left(\tau^5\right)$ hibája mellett, ahol τ a lépéshossz), addig a léptető szabálya miatt - mely a következő:

$$\vec{R}_{n+1} = 2\vec{R}_n - \vec{R}_{n-1} + \tau^2\vec{A}_n + \mathscr{O}\left(\tau^4\right) \qquad (1)$$

$$\vec{V}_n = \frac{\vec{R}_{n+1} - \vec{R}_{n-1}}{2\tau} + \mathscr{O}\left(\tau^2\right) \tag{2}$$

nem indítható egy tetszőleges kezdeti feltételből: a helykoordináták léptetése két előző pontot használ fel, így azokat a szimuláció elején már ismerni kell. Emellett $\vec{V}$-ben csak $\mathscr{O}\left(\tau^2\right)$ pontosságú. Ezzel ellentétben a Velocity–Verlet-algoritmus $\vec{R}$-ben és $\vec{V}$-ben egyaránt csak $\mathrm{O}\left(\tau^3\right)$ hibával rendelkezik, de cserébe indítható egy általunk választott kezdőpontból. Léptető szabálya a következő:

$$\vec{R}_{n+1} = \vec{R}_n + \tau\vec{V}_n + \frac{\tau^2}{2}\vec{A}_n + \mathscr{O}\left(\tau^3\right) \tag{3}$$

$$\vec{V}_{n+1} = \vec{V}_n + \frac{\tau}{2}\left(\vec{A}_{n+1} + \vec{A}_n\right) + \mathscr{O}\left(\tau^3\right) \tag{4}$$

Ha csak a koordináták pontosságát és az energiamegmaradást tartjuk fontosnak, akkor megoldást nyújthat a kettő kombinálása, ahol az első két pontot a Velocity–Verlet-módszerrel, onnantól kezdve pedig mindet a Verlet-algoritmussal határozzuk meg. Itt azonban most az egyszerűség kedvéért mindenhol a Velocity–Verlet-módszert alkalmaztam.

3.2. LENNARD–JONES-POTENCIÁL

A szimulációban van der Waals-közelítést használunk, mely esetén az egyástól r távolságra levő részecskék közti kölcsönhatást a Lennard–Jones-potenciál[3] írja le:

$$V(r) = 4V_0\left[\left(\frac{r_0}{r}\right)^{12} - \left(\frac{r_0}{r}\right)^{6}\right] \tag{5}$$

Ennek alakja könnyen felírható. A potenciál képletében az r^{-12} tag a Pauli-féle kicserélődési kölcsönhatás miatt fellépő faktor, mely rövid távolságokon érvényesül és nagyon erős taszító erőt fejt ki. Míg az r^{-6} tag a nagyobb távolságokon fellépő van der Waals-erők miatt jön a képletbe, melyek itt vonzó erőt hoznak létre. Egy karakterisztikus $r_m = 2^{1/6}r_0$ pontban a függvény eléri a minimumát, a vonzó erő itt a legnagyobb. Innentől r-t csökkentve a vonzás szintén csökken, majd egy adott ponton eléri a nullát. Ha a részecskék tovább közelednek még ezután, akkor egy taszító erő lép fel, amely rendkívül gyorsan növekszik, és így a részecskéket ellöki egymástól.
Maga az erőt, mellyel a léptetési szabályokban felhasznált $\vec{A}$ gyorsulásokat megadja, a potenciál gradiensében számíthatjuk legegyszerűbben:

$$\vec{F}(\boldsymbol{r}) = -\boldsymbol{\nabla}V(r) = \frac{24V_0}{r^2}\left[2\left(\frac{r_0}{r}\right)^{12} - \left(\frac{r_0}{r}\right)^{6}\right]\boldsymbol{r} \tag{6}$$

3.3. MÉRT MENNYISÉGEK

Több különböző, a (2) részben már felsorolt mennyiség értékét kellett mérjük a szimuláció során. Hogy ezeket numerikusan vizsgálni tudjuk, javasolt volt számunkra úgy megválasztani az egyes összefüggésekben szereplő karakterisztikus mennyiségeket, hogy azok értékei mind 1-et vegyenek fel. Így a következő mértéket választottam a feladatok megoldása során:

$$V_0 = r_0 = m = 1 \tag{7}$$

Mivel a hőkapacitás kiszámításához szükségünk volt a teljes energia - közelítésben csak - időátlagára, ezért előbb az energiát kellett kiszámolnunk. A teljes energia egyszerűen felírható a Hamilton-függvény segítségével:

$$\mathscr{H} \equiv E = \frac{m}{2}\sum_{i=1}^{N}\boldsymbol{v}_i^2 + \sum_{i\neq j}V\left(|\boldsymbol{r}_i - \boldsymbol{r}_j|\right) \tag{8}$$

Ahol $V\left(|\boldsymbol{r}_i - \boldsymbol{r}_j|\right)$ az i és j indexű részecskék közti Lennard–Jones-potenciál, $\boldsymbol{v}_i$ pedig az i-edik részecske sebessége.
Másodikként a moláris, konstans térfogaton mért hőkapacitást határoztam meg, melyet a gáz fundamentális egyenletéből kaphatunk a következő módon:

$$C_V = \left(\frac{\partial E}{\partial T}\right)_V = \frac{1}{k_B T^2}\left[\left\langle E^2\right\rangle - \langle E\rangle^2\right] \tag{9}$$

Ahol k_B a Boltzmann-állandó. Ezen fenti mennyiség mértékegysége $\frac{J}{K\cdot mol}$, nagyságrendje standard körülmények között általában $10^1 - 10^2$ között található[4]. Az

$$\left[\left\langle E^2\right\rangle - \langle E\rangle^2\right] \tag{10}$$

szórásban szereplő átlagokat időátlagokként közelítettem, ahogy az a feladat kiírásában is javasolva volt[5].
Végezetül a nyomást és a kompresszibilitási faktort a Viriál-tétel segítségével mértem, melyből felírható az alábbi összefüggés:

$$PV = Nk_BT + \frac{1}{3}\left\langle\sum_{i<j}r_{ij}\cdot F_{ij}\right\rangle \tag{11}$$

Melyből a kompresszibilitási faktor:

$$Z = \frac{PV}{Nk_BT} \tag{12}$$

Mely utóbbinak értéke ideális gázra $Z = 1$, míg nagy sűrűségű közegre $Z > 1$, kis sűrűség esetén pedig $Z < 1$.

4. MEGVALÓSÍTÁS

Előzetesen az (2)-es pontban már tisztázott funkcionalitású, `C++` nyelven írt molekuladinamikai szimulációt megvalósító keretrendszerek voltak számunkra megadva, melyeket szintén a (2)-es pontban ismertetett szempontok alapján, nekünk kellett bővítenünk.
A forráskódot az eddigiekhez hasonlóan egy saját batch file segítségével, benne a `clang` fordító felhasználásával fordítottam. Az eredeti kód módosításával elértem, hogy a lefordított `exe` program egy Jupyter Notebook-ban futó Python 3 kernel segítségével induljon. A szimuláció a kezdőfeltételeket szintén ebből a környezetből várja, bemenő paraméterek formájában.

Az eredeti integritást meghagyva, a három különböző szinten megvalósított MD szimuláció három `main` forrásfile alapján fordul. Az egyes lefordított `exe`-ket a kezdőfeltételek mellett megadott másik bementei paraméter segítségével lehet tetszőlegesen keményfalú, periodikus, vagy határfeltétel nélküli módokban lefuttatni. Ezek kimenete minden szimuláció esetében egy-egy `.dat` file, melyek minden sora egy-egy szimulációs lépésnek felel meg. Egy sor minden esetben tartalmazza az összes szimulált részecske 3 térkoordinátáját, az azokhoz tartozó sebesség- és gyorsuláskomponenseket, valamint sorrendben az összes többi vizsgálandó mennyiséget az adott lépés esetére. Ezek pontos sorrendjéért lásd a programkódot GitHub-on[6].
Az előző szimulációkhoz hasonlóan, most is készítettem néhány animációt a szimulált folyamatokról. Szintén az előzőekhez hasonlóan, ezt most is Pythonban valósítottam meg egy saját kód segítségével, ami a `matplotlib` és a `imageio` könytárakat használva generál megadott paraméterek alapján `mp4` formátumú, kis méretű, de nagy felbontású videókat. Viszont ellentétben az eddigiekkel, most minden részfeladatról animációt készítettem, amik válogatva a YouTube-on megtekinthetőek[2].
A végleges forráskódok és a programokat futtató Notebook file szintén mind elérhető GitHub-on[6].

5. KIÉRTÉKELÉS

6. FUTÁSIDŐ

1. ábra. Placeholder

7. DISZKUSSZIÓ

[1] Loup Verlet. “Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules”. In: *Physical review* 159.1 (1967), p. 98.

[2] *Pál Balázs's Channel — YouTube*. [Online; opened at March 23, 2019]. 2019. URL: https://www.youtube.com/channel/UCBDSB7PdQ3E9l9WSBsTy7cQ.

[3] J. E. Jones. “On the Determination of Molecular Fields. II. From the Equation of State of a Gas”. In: *Proceedings of the Royal Society of London Series A* 106 (Oct. 1924), pp. 463–477. DOI: 10.1098/rspa.1924.0082.

[4] Stephen T. Thornton and Andrew Rex. *Modern physics for scientists and engineers*. Cengage Learning, 2012.

[5] József Stéger, István Csabai. *Számítógépes szimulációk – Molekuladinamika*. [Online; opened at March 25, 2019]. 2019. URL: https://stegerjozsef.web.elte.hu/teaching/szamszim/moldin.pdf.

[6] Pál Balázs. *ELTE Computer Simulations 2019 — GitHub*. [Online; opened at March 23, 2019]. 2019. URL: https://github.com/masterdesky/ELTE_Comp_Simulations_2019.