SZÁMÍTÓGÉPES SZIMULÁCIÓK

Molekuladinamika
jegyzőkönyv

Jegyzőkönyvet készítette:
Koroknai Botond (AT5M0G)

Jegyzőkönyv leadásának időpontja:
2024.05.09

1. Probléma rövid ismertetése

A feladat során nagyszámú (nagyságrendileg $6 \cdot 10^{23}$) részecske dinamikáját szeretnénk vizsgálni. Mivel ez nagy számításkapacítást igényelne különféle közelítésekkel élhetünk. Ahol az erő gyorsan lecseng ott felthetjük, hogy:

- a távoli erők nagyobb térrészekre kiátlagolhatóak
- a nagyon távol eső részecskék pedig elhanyagolhatóak.

A vizsgált probléma során N darab atomunk van, melynek ismerjük a kezdő sebességét, valamint kezdő pozícióját. A részecséket az időben fogjuk léptetni, és a köztük ható erőket páronként fogjuk számolni.

Azt szeretnénk vizsgálni, hogy bizonyos idő elteltével, vajon egyensúlyi állapotba kerül-e a rendszerünk? Ugyan is az egyensúlyi állapotról azt tudjuk, hogy ott teljesülni kell az ekvipartíció tételének:

$$\langle K \rangle = \left\langle \frac{1}{2} m v^2 \right\rangle = \frac{3}{2} k_B T \tag{1}$$

Ha igazán pontos szimulációt szeretnénk kapni, akkor az atomok belső szerkezetét is figyelembe vevő kvantummechanikát kellene használni, azonban nemesgázok esetén a klasszikus részecske közelítés is elfogadható eredményre vezet, mert:

- $T = 300K$ hőmérsékletű Ar gázban az egy részecskére jutó mozgási energia

  $$\frac{3}{2} k_B T \approx 6.2 \cdot 10^{-21}\, J \approx 0.039\, eV$$

  Ami jóval kisebb mint az elektronok 11.6 eV - os gerjesztési energiája.

- Valamint az $m = 6.69 \cdot 10^{-26}$ tömegű Ar atomok de Brogille-hullámhossza:

  $$\frac{2\pi\hbar}{p} = \frac{2\pi\hbar}{\sqrt{3 m k_B T}} \approx 2.2 \cdot 10^{-11}\, m$$

  kisebb mint az atomok mérete, és így az atomok közti típikus $r_0 = 3.4 \cdot 10^{-10}$ távolságnál is.

Mivel a szimuláció így is idő igényes fontos, hogy olyan kezdeti feltételek állítsunk be, amik egy termodinamikai egyensúly közeli állapotot írnak le.

A molekuladinamika szimulálására a két legelterjedtebb differenciálegyenlet léptető eljárás a *Verlet-alogritmus* és a *velocity-Verlet-alogrimtus*.

A részecskéink közti kölcsönhatásokat van der Waals közelítésben a Lenard-Jones potenciállal írhatjuk le:

$$V(r) = 4V_0 \left[\left(\frac{r_0}{r} \right)^{12} - \left(\frac{r_0}{r} \right)^{6} \right] \tag{2}$$

Ahol r az atomok közötti távolság és r_0 a már korábban ismertetett atomok közötti típikus távolságot jelöli. Továbbá az erőt is megkaphatjuk a Lenard-Jones potenciál negatív gradienséből, így a rendszer energiája a

$$E = \sum_{i=1}^{N} \frac{1}{2} m v_i^2 + \sum_{\langle ij \rangle} V(r_{ij}) \tag{3}$$

képletből adódik, és r_{ij} jelöli $\langle ij \rangle$ atompár távolságát. Egyensúlyban a sebességek a Maxwell-Boltzmann eloszlást követeik.

Amit még fontos megjegyezni, hogy 3D-s szimuláció során periodikus határfeltételeket kell alkalmaznunk, valamint a potenciális energiaminimum miatt célszerű a rendszert lapcentrált köbös elrendezésből indítani. Ebben az esetben a Gauss-Boltzmann eloszlás alakja:

$$P(v) = \left(\frac{m}{2\pi k_B T} \right)^{\frac{3}{2}} e^{-\frac{m(v_x^2 + v_y^2 + v_z^2)}{2 k_B T}} \tag{4}$$

Mivel a program legidőigényesebb része a páronkénti erők kiszámítása, használjuk a L. Verlet által ismertetett két gyorsítást:

- Potenciál levágás: Mivel az LJ potenciál rövid hatótávolságú, ezért bevezethető egy úgynevezett r_{cutoff} levágási táv, ami felett az erő hatását zérusnak vesszük.
- Szomszédsági lista: A levágáshoz tudnunk kell mely párokra teljesül, hogy: $r_{ij} = |r_i - r_j| < r_{cutoff}$, csakhogy minden részecske mozog. Vegyük figyelmbe, hogy egy-egy lépés során a részecskék nagyon kicsit mozdulnak el, így választhatunk egy olyan távolásgot, aminél jobban biztos nem távolodnak el pár lépés alatt. Ennek segítségével számon tudjuk tarttani a szomszédban lévő atomokat és a listát csak néha frissítenünk.

Ezen javaslatok hatására a program futásideje jelentősen csökkenhet, viszont sértjük az energiamegmaradást, amit a következő képpen korrigálhatunk:

$$V_{corr}(r) = V(r) - \left.\frac{dV}{dr}\right|_{r_{cutoff}} (r - r_{cutoff}) \tag{5}$$

1.1. Termodinamikai mennyiségek

1.1.1. Hőmérséklet

Mint azt korábban ismertettem termodinamikai egyensúlyban érvényesül a ekvipartíció tétele, és a kinetikus energiából számolhatunk hőmérsékletet:

$$3(N-1) \cdot \frac{1}{2} k_B T = \left\langle \frac{m}{2} \sum_{i=1}^{N} v_i^2 \right\rangle \tag{6}$$

ahol $3(N-1)$ a belső szabadsági fokok száma.

1.1.2. Teljes energia

A mozágsi energia és potenciális energia összegeként kapjuk meg:

$$E = \frac{m}{2} \sum_{i=1}^{N} \mathbf{v}_i^2 + \sum_{i \neq j} V(|\mathbf{r}_i - \mathbf{r}_j|) \tag{7}$$

1.1.3. Hőkapacítás

$$C_v = \left(\frac{\partial E}{\partial T}\right)_V = \frac{1}{k_B T^2} \left[\langle E^2 \rangle - \langle E \rangle^2\right] \tag{8}$$

Több különböző véletlen futtatás eredménye, de egyensűlyban ezt jól közelíti az időátlag. A hőkapacítást továbbbá a hőmérséklet-fluktuációkból is következtethetjük:

$$\langle T^2 \rangle - \langle T \rangle^2 = \frac{3}{2} N (k_B T)^2 \left[1 - \frac{3N k_B}{2C_V}\right] \tag{9}$$

1.1.4. Nyomás

A Viriál-tétel alapján a nyomás

$$PV = N k_B T + \frac{1}{3} \left\langle \sum_{i<j} \vec{r_{ij}} \cdot \vec{F_{ij}} \right\rangle \tag{10}$$

1.1.5. Kompresszibilitás faktor

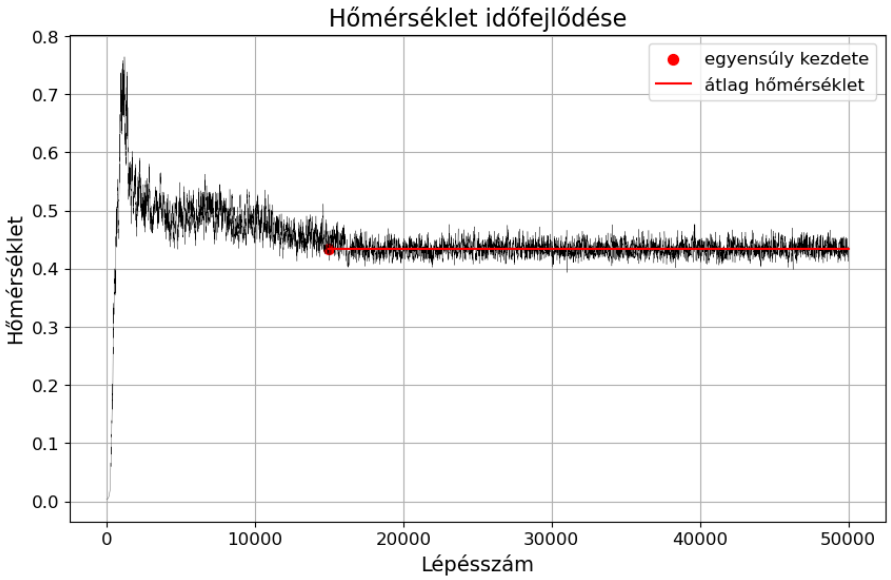
Kifejezi, hogy mennyire vagyunk távol az ideális gáz állapottól:

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

2. Feladatok

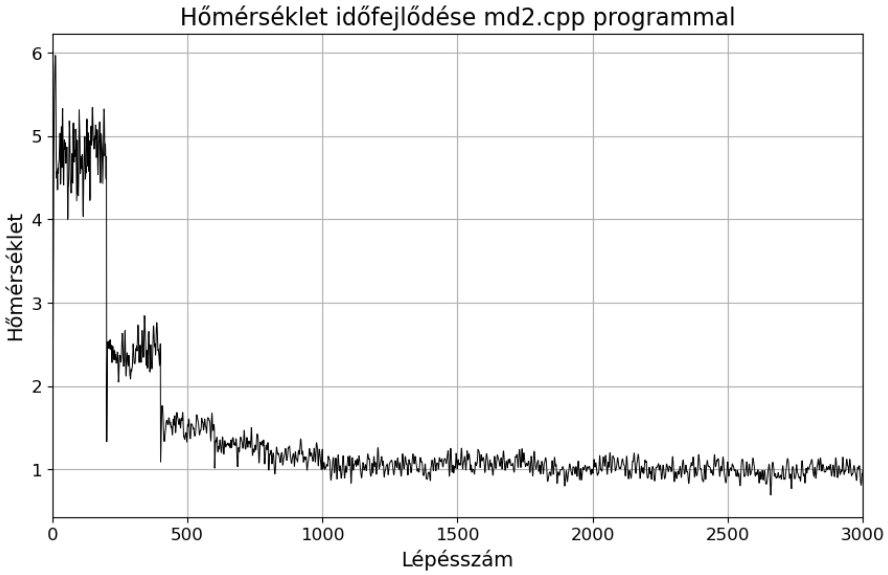
2.1. 1. feladat

Az első feladat az md.cpp program megértése volt. A feladat egy szimulációt hajt végre a *velocity-Verlet-aglortimust* segítségével, és ”T.data” nevű fileba kimenti minden egyes idő pillanatban a rendszer pillanatnyi hőmérsékletét. A kódot első probálkozásra 10 ezer lépéssel futattam, de ekkor még nem tapasztaltam egyensúlyi helyzetet, úgyhogy megismételtem a szimulációt 50 ezer lépéssel is.



1. kép. A szimuláció 50 ezer lépésre futtatva

Láthatjuk, hogy az egyensúly körülbel 15 ezer lépést követően áll be, ugyan is ekkor már jelentős változást nem tapasztalunk, csupán egy adott hőmérsékleti sávban kicsi ingadozásokat. A feladat második része a md2.cpp programban eszközölt javítások vizsgálata volt.



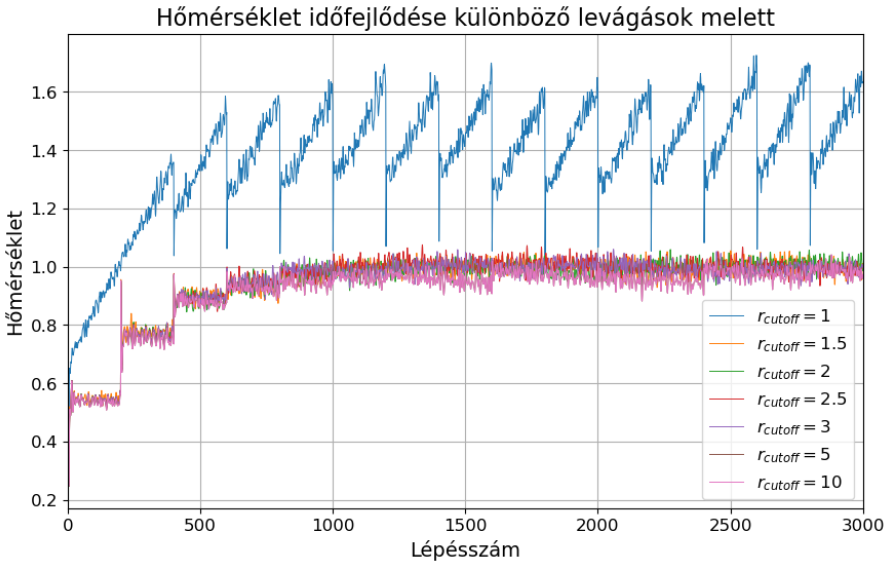
2. kép. Javíott szimuláció 50 ezer lépésre futtatva

A főbb módosítások a legelső kódhoz képest az, hogy: figyelembe vettük a periódikus határfeltételeket, minden részecske számára inicializál kezdőfeltételt a *gasdev()* függvény segítségével, amely a Gauss-eloszlás szerint ad visssza véletlen értékeket, majd ezeket elosztja a három irányba, közben figyelembe véve, hogy a rendszer súlyponti sebessége nulla maradjon. Továbbá a a sebességek újra skálázását is elvégzi, hogy megkapjuk a kívánt pillanatnyi hőmérsékletet, amit a kód elején a T paraméterrel tudunk beállítani.

Ami egyből szembetűnő a második szimuláció futtatásakor az a gyorsaság, ebben az esetben körülbelül csak ezer lépésre volt szükség ahhoz, hogy az egyensúly beálljon, a korábbi 15 ezerrel szemben.

2.2. 2. feladat

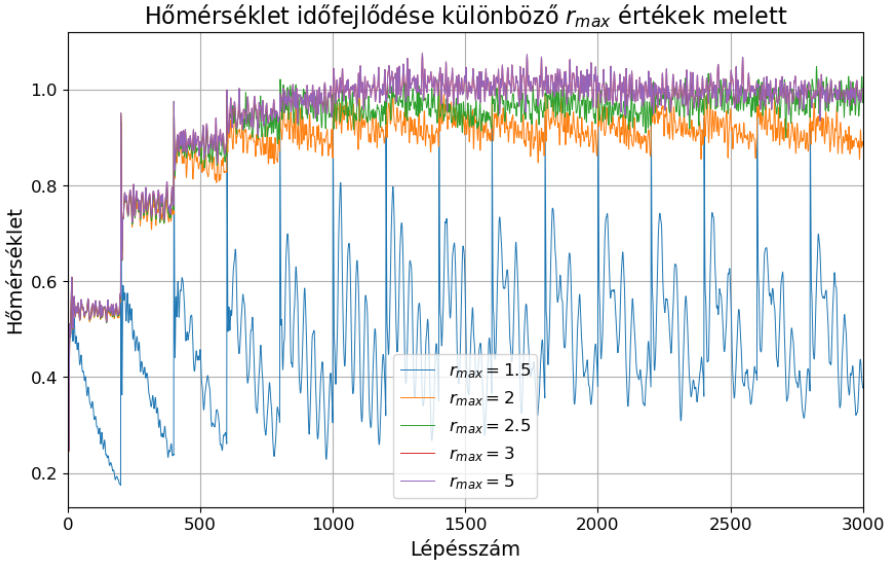
A feladat során r_{cutoff}, r_{max} és *updateInterval* paraméterekkel kellett, hogy kísérletezzünk, hogy megértsük miként kezeli a program a levágásokat, a szomszédsági listát, és annak frissítését.



3. kép. md3.cpp 3 ezer lépésig futtatva

Mint ahogy a probléma ismertetésénél megírtam, Verlet által javasolt érték az $r_{cutoff} = 2.5\, r_0$. Ezt az értéket felülről, és alulról és körbejártam és azt tapasztaltam, hogy ha növeljük az értékét, akkor lényeges változást nem tapasztalunk. Ha viszont elkezdjük csökkenteni, egy bizonyos érték alá lépve, instabillá válik a hőmérséklet.

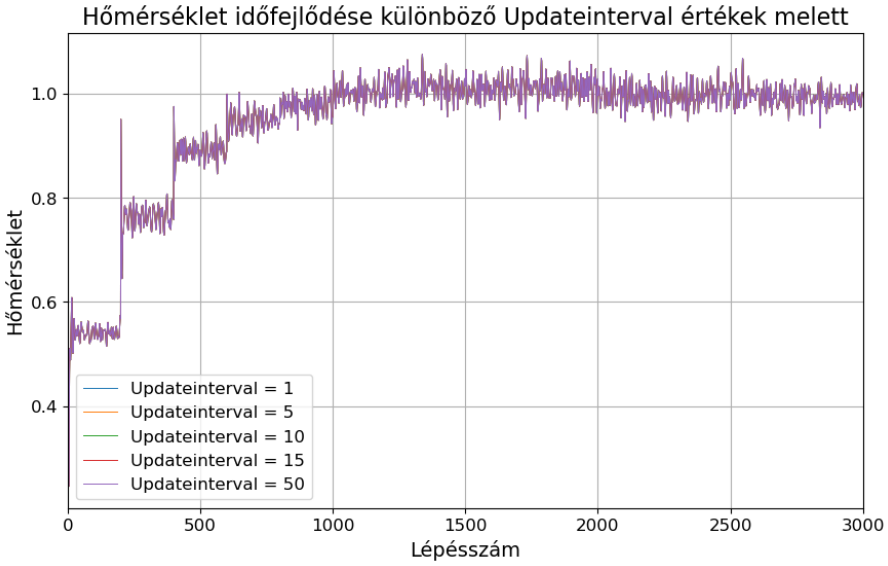
Ezt követte az r_{max} paraméter értékének változtatgatása:



4. kép. md3.cpp 3 ezer lépésig futtatva

Azt tapasztaltam, hogy amennyiben elkezdjük csökkenteni az r_{max} paraméter értékét, akkor csökken a rendszer egyensúlyi hőmérséklete, valamint a hőmérséklet ingadozás egyre nagyobb lesz.

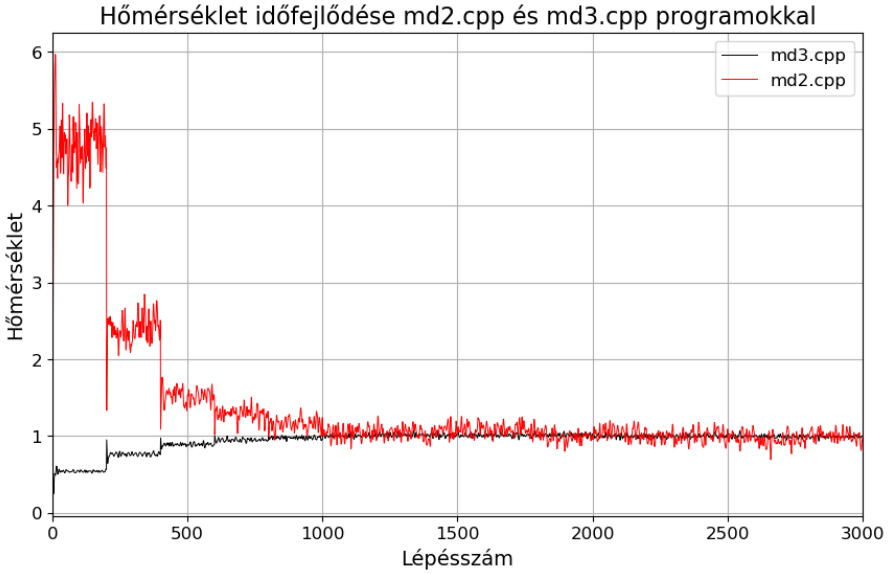
Végül pedig az *Updateinterval* paraméter módosítására bekövetkező hatásokat vizsgáltam meg:



5. kép. md3.cpp 3 ezer lépésig futtatva

Ebben az esetben semmilyen változást nem tapasztaltam.

Ezt követően összehasonlítottam a *md2.cpp* és *md3.cpp* programokat:



Mind sebesség (kevesebb lépés alatt egyensúlyban), mind egyensúly (kisebb fluktuáció) szempontjából a md3.cpp bizonyult pontosabbnak.

3. Megjegyzés:

Az elméleti összefoglalót a fealadathoz rendelt diasor alapján készítettem el.