Molekuladinamika

Márton Tamás

Eötvös Lóránd Tudományegyetem, Informatikus Fizikus

Számítógépes szimulációk laboratórium.

V. jegyzőkönyv.

Tartalomjegyzék

Ábrák jegyzéke

1. Fizikai probléma ismertetése

Molekuladinamika alatt nagy számú részecske mikroszkopikus dinamikájának vizsgálatát értjük. Mivel viszont a részecskeszám akár $6 \cdot 10^{23}$ nagyságrendű is lehetne, ezért még a mai számításkapacitás mellett is közelítéseket alkalmazunk ekkora rendszereknél, ahol az erő gyorsan lecseng:

- a messzi erők kiátlagolódnak nagy térrészekre,
- azok a részecskék, amik nagyon távol esnek, elhanyagolhatóak.

A probléma során adott N darab atomunk, akiknek ismerjük a kezdő időpontban a helyzetét és a sebességét. Az egyes részecskék között páronként számolunk erőket, és így léptetjük időben a szimulált rendszert, és azt vizsgáljuk, hogy valamennyi idő eltelte után egyensúlyba konvergál-e a rendszerünk. Az egyensúlyról tudjuk, hogy ott teljesülni kell az ekvipartíció tételének, azaz:

$$\langle K \rangle = \left\langle \frac{1}{2} m v^2 \right\rangle = \frac{3}{2} k_B T. \tag{1.1}$$

A precizitás kedvéért mondhatnánk azt, hogy használnunk kele a kvantummechanikát, de beláthatjuk, hogy ha nemesgázokat szimulálunk, akkor a klasszikus részecske közelítés jó képet nyújt számunkra, hiszen:

- $T = 300K$ szobahőmérsékletű Ar gáz esetében az egy részecskére jutó mozgási energia $(3/2 k_B T \approx 0.039eV)$ az elektronok gerjesztési energiájának $(11.6eV)$ kb. 0.3%-a csupán,
- az Ar atomok tömege $m = 6.69 \cdot 10^{-26} kg$, így a de Broglie-hullámhossza: $\lambda = \frac{2\pi\hbar}{\sqrt{3mk_BT}} \approx 2.2 \cdot 10^{-11} m$, ami kisebb, mint az atom effektív mérete, vagyis az atomok között lévő tipikus távolságoknál is kisebb.

A szimuláció során azon problémás lehet a termodinamikai egyensúly elérése, mert nagyon kicsi időközönként kellene nagyon sok atomra kiszámolni az erőket. Így arra kell törekednünk, hogy a szimuláció kezdeti feltételeit a termodinamikai egyensúly közelébe próbáljuk beállítani.

2. Szimuláció

Molekuladinamika szimulációkra a legelterjedtebb differenciálegyenlet léptető eljárások:

- Verlet-algoritmus
- velocity-Verlet-algoritmus

A szimulálni kívánt részecskéink között lévő kölcsönhatást a Lennard–Jones-potenciállal tudjuk leírni avan der Waals-közelítésben:

$$V(r) = 4V_0 \left[\left(\frac{r_0}{r} \right)^2 - \left(\frac{r_0}{r} \right)^6 \right] \tag{2.1}$$

ahol r az atomok középpontjai közötti távolságot, r_0 pedig az atomok közötti tipikus távolságot ($r_0 = 3.4 \cdot 10^{-10} m$) jelöli. A Lennard–Jones-potenciál a következő tulajdonságokkal rendelkezik:

- előjelet vált, ha $r = r_0$,
- minimuma $-V_0$, amit $r = 2^{1/6} r_0$ érték mellett vesz fel,
- az általa leírt erő erősen taszít kicsi távolságokban: zárt elektronhéjak Pauli-féle kizárási elv,
- az általa leírt erő gyengén vonzó nagy távolságokon: indukált elektromos dipólerők a van der Waals-kölcsönhatás miatt.

A Lennard–Jones-potenciál negatív gradiensébőlmegkaphatjuk az erőt, s a teljes szimulált rendszer energiája:

$$E = \sum_{i=1}^{N} \frac{1}{2} m v_i^2 + \sum_{\langle ij \rangle}^{V(r_{ij})} . \tag{2.2}$$

Ahol r_{ij} jelöli az $\langle ij \rangle$ atompár közötti távolságot. Egyensúly esetén a sebességeknek a Maxwell–Boltzmann-eloszlást kell követniük. Ha N darab Ar atomot szeretnék vizsgálni egy négyzetben (2D), akkor megadhatjuk, hogy:

- a 2 formulában szereplő $V_0 = 1.65 \cdot 10^{-21} J$, vagyis $V_0/k_B = 119.8K$ illetve $r_0 = 3.41 \cdot 10^{-10} m$,
- a rendszerünk q karakterisztikus ideje, ami az atomok közötti átlagos távolság megtételéhez szükséges idő: $\tau_0 = \sqrt{\frac{mr_0^2}{V_0}} = 2.17 \cdot 10^{-12} s$.

Mivel látjuk, hogy ezek az értékeke eszméletlen picik **SI** mértékegységben, a numerikus számábrázolás tulajdonságai miatt inkább egy olyan skálarendszert kell alkalmaznunk, ahol:

$$\boldsymbol{m = 1,}\ r_0 = 1, V_0 = 1, \tau_0 = 1.$$

Háromdimenziós szimuláció esetén a az energiaminimum miatt a rendszerünket lapcentrált köbös elrendezésből indítva, periodikus határfeltétellel modellezhetjük, s 3D-ben a Maxwell–Boltzmann-eloszlás a Gauss-eloszlás:

$$P(v) = \left(\frac{m}{2\pi k_B T}\right)^{3/2} exp(-\frac{m(v_x^2 + v_y^2 + v_z^2)}{2k_B T}) \tag{2.3}$$

A szimulációs program legidőigényesebb része természetesen az atompárok között fellépő erők kiszámítása. Erre L. Verlet cikkében két gyorsítást talált ki:

1. **Potenciális levágás**:mivel a Lennard–Jones-potenciál által leírt erő rövid hatótávolságú, $r > r_0$ felett gyorsan lecseng, ezért bevezethető egy r_{cutoff} levágási táv, ami felett az erő hatását zérusnak vesszük, és ez akkor hasznos, ha $r_{cutoff} << L$ (L a szimulált rendszer lineáris mérete).
2. **Szomszédsági lista**: a levágáshoz tudnunk kell, hogy mely párokra igaz az, hogy $r_{ij} = |r_i - r_j| < r_{cutoff}$, de ugye minden részecskénk mozog. Ha azt vesszük, hogy az atomok az egyes lépéseknél csak kicsit mozdulnak el, választhatnánk egy olyan $L >> r_{max} > r_{cutoff}$ távot, aminél jobban nem távolodnak el az atomok pár lépés alatt, és így nyilvántartjuk a szomszédjukban lévő atomokat, és csak néha frissítjük azt. Verlet azt javasolta, hogy:
   - $r_{cutoff} = 2.5 r_0$
   - $r_{max} = 3.2 r_0$
   - 10 lépésenként frissítsük a szomszédsági listát.

Verlet módosításai miatt a program futásideje jóval csökkenhet, ám a az energiamegmaradás már nem igaz, de korrigálható a 2. formula:

$$V_{corr}(r) = V(r) - \left.\frac{dV}{dr}\right|_{r_{cutoff}} (r - r_{cutoff}) \tag{2.4}$$

3. Mérhető termodinamikai mennyiségek

Hőmérséklet:

Mint már említettem, egyensúlyban az ekvipartíció tétele teljesül, s a mozgási energiából tudunk számolni hőmérsékletet, s mivel a Lennard–Jones-erő konzervatív, az energia állandó:

$$3(N-1)\cdot\frac{1}{2}k_BT=\left\langle\frac{m}{2}\sum_{i=1}^{N}v_i^2\right\rangle. \tag{3.1}$$

Ahol *3(N - 1)* a belső szabadsági fokok száma. Viszont fontos megjegyeznünk, hogy a tömegközéppont mozgása nem ad járulékot a termikus energiába, ha az átlagsebesség v nem zérus, ezértakkor v_i^2 helyett $(v_i-v)^2$ kerül az energiába. Természeten ha nincsen adott pillanatban egyensúly, az ekvipartíció nem érvényes, de az ebből a kifejezésből számított hőmérséklet az adott időponthoz hőmérsékletet megadja nekünk.

Teljes energia:

egyszerűen csak a mozgási és a potenciális energia összege:

$$E=\frac{m}{2}\sum_{i=1}^{N}\vec{v}_i^2+\sum_{i\neq j}V(|\vec{r}_i-\vec{r}_j|). \tag{3.2}$$

Hőkapacitás:

az átlagolások sokaságátlagok, azaz több különböző szimulációból átlagosan kapott eredmény, de ha egyensúlyba relaxált a rendszer, akkor ez jó közelítés:

$$C_V=\left(\frac{\partial E}{\partial T}\right)_V=\frac{1}{(k_BT)^2}\left[\langle E^2\rangle-\langle E\rangle^2\right], \tag{3.3}$$

azonban ezt hőmérséklet fluktuációkból is megbecsülhetjük:

$$\langle T^2\rangle-\langle T\rangle^2=\frac{3}{2}N(k_BT)^2\left[1-\frac{3Nk_B}{2C_V}\right]. \tag{3.4}$$

Nyomás:

A Viriál-tétel alapján a következő formulából mérhetjük a nyomást:

$$PV = Nk_BT + \frac{1}{3}\left\langle \sum_{i<j} \vec{r}_{ij} \cdot \vec{F}_{ij} \right\rangle, \tag{3.5}$$

de megmérhetnénk a doboz falánál történő ütközések impulzusváltozásából is.

Kompresszibilitás:

Kifejezi, hogy mennyire vagyunk távol az ideális gáz állapottól:

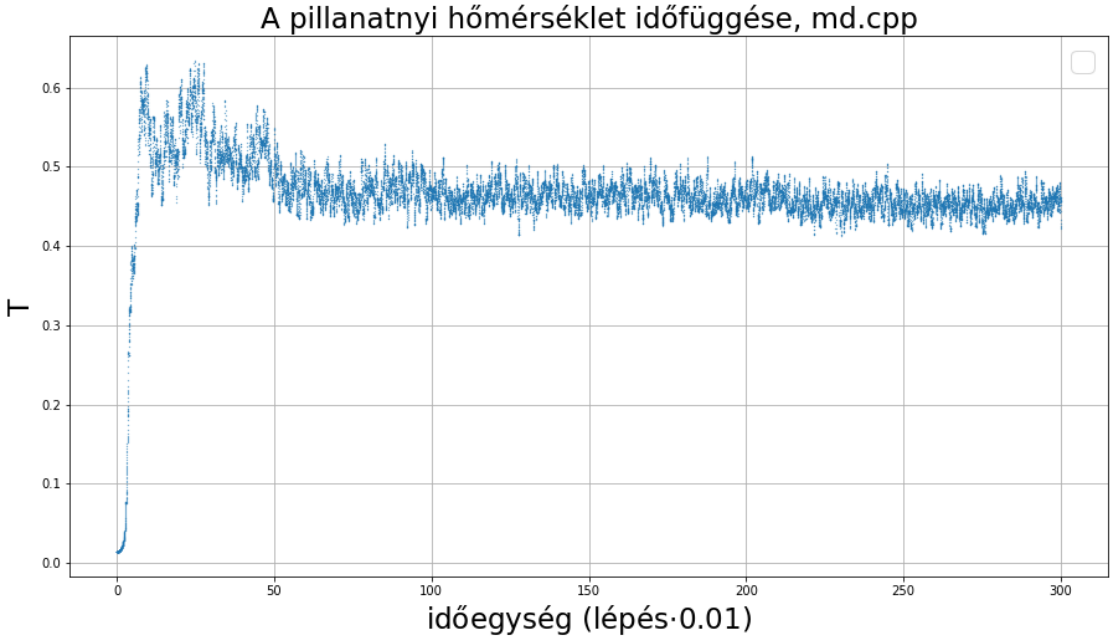
$$Z = \frac{PV}{Nk_BT} = 1 + \frac{1}{3Nk_BT}\left\langle \sum_{i<j} \vec{r}_{ij} \cdot \vec{F}_{ij} \right\rangle. \tag{3.6}$$

A taszító potenciál miatt nagy sűrűségen $Z > 1$, kisebb sűrűségen a vonzó van der Waals-erők miatt $Z < 1$. A $Z = 1$ eset felel meg szabad részecskének, ideális gáznak.

4. Megoldások, Eredmények értelmezése

1.feladat

Első lépés az *md.cpp* program megértése volt a cél. Nos, ez a program a sima velocity-Verlet-algoritmus alapján szimulál, és minden egyes részecskepárra számol erőt, kimeneti file-ja pedig a rendszer pillanatnyi hőmérséklete minden egyes szimulációs lépés után. Példaként lefuttattam a szimulációt ***30000*** léptetésre, és kiábrázoltam a hőmérsékletet, ezt láthatjuk a 4..1. ábrán. Jól látszik, hogy olyan ***5000*** léptetés után beáll a rendszer az egyensúlyba, hiszen utána egy megadott hőmérséklet sávban fog fluktuálni.

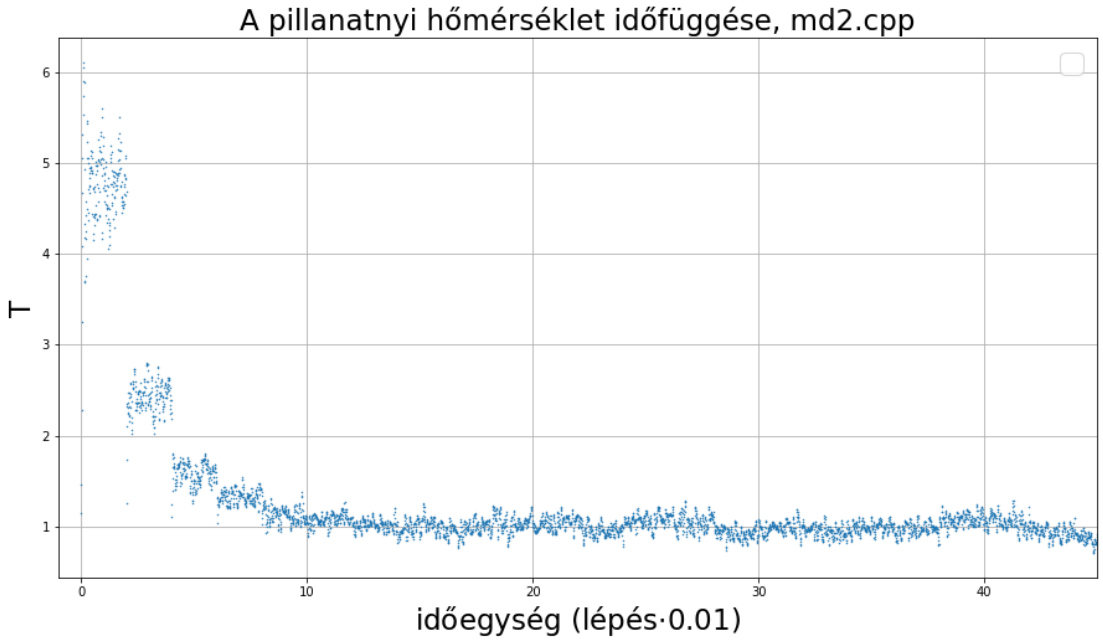


4..1. ábra. md.cpp által kiszámolt pillanatnyi hőmérsékletek az egyes léptetésekre.

A következő lépés az volt, hogy értsük meg, mi a különbség az *md.cpp* és az *md2.cpp* programok között, így megpróbálom pontokba szedni az *md2.cpp* program fejlesztéseit az *md.cpp*-hez képest:

- Tartalmaz periodikus határfeltételt.
- A kezdősebesség beállítása a *gasdev()* függvénnyel történik, ami egy 0 várható értékű, 1 szórású számot generál, ezután a sebességeket korrigálja a tömegközéppont sebességével, majd a diasoron említett módon átskálázza a sebességeket, hogy megkapjuk a pillanatnyi hőmérsékletet.
- A programkód elején szereplő $T = 1$ hőmérséklet az a paraméter, amihez konvergálni fog a hőmérséklet az egyensúlyban.

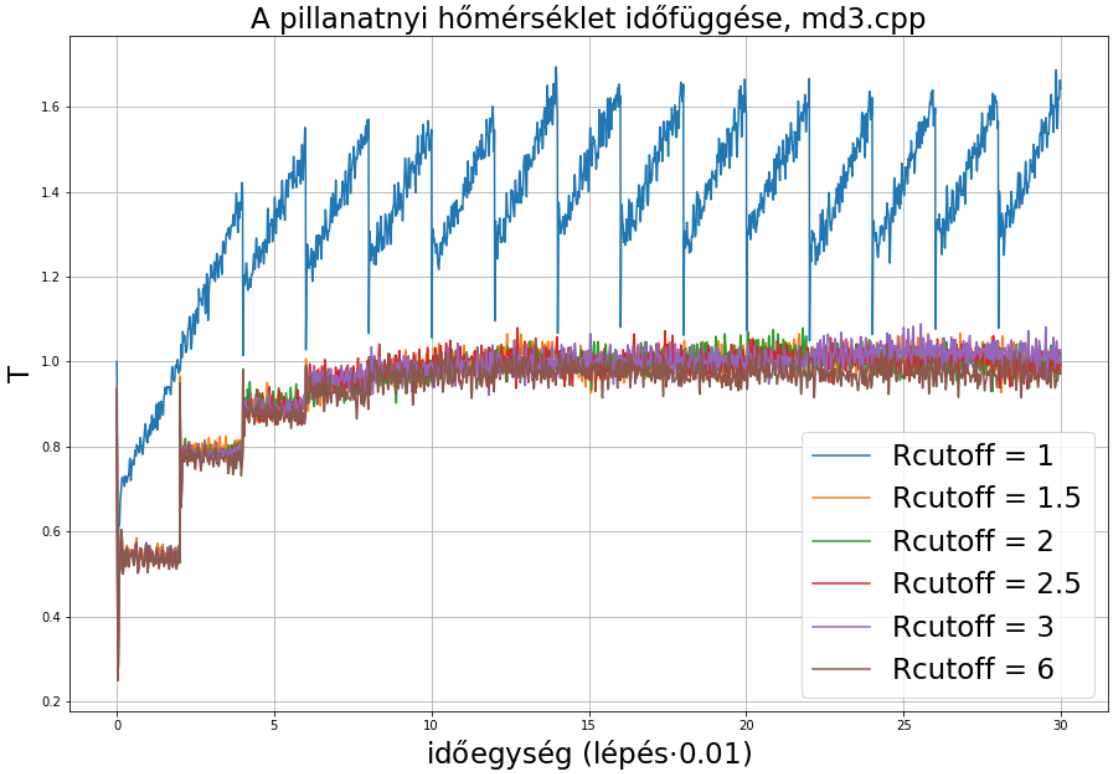
A 4..2. ábrán láthatjuk az *md2.cpp* által szimulált rendszer pillanatnyi hőmérsékletét a léptetések függvényében. Jól látszik, hogy az 4..1. ábrával ellentétben, itt sokkal gyorsabban beáll az egyensúlyi hőmérséklet, mégpedig olyan ***1000***lépés alatt körülbelül, ami az *md.cpp* ***5000*** lépéséhez képest jóval kisebb.



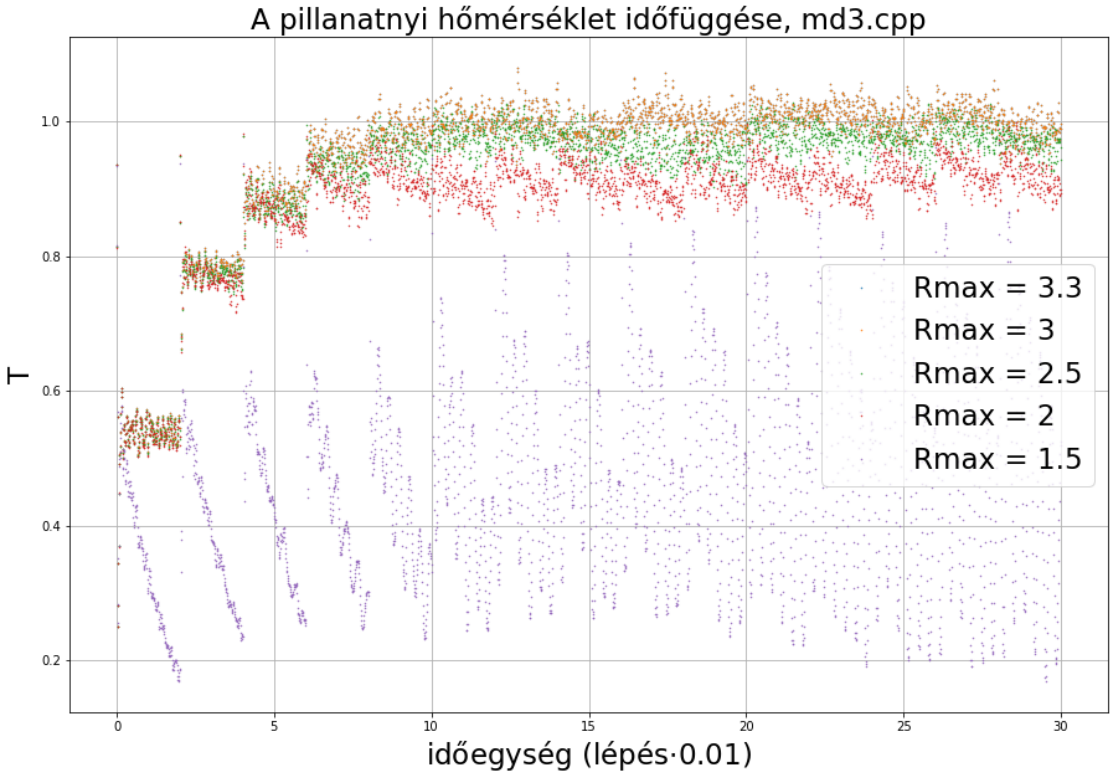
4..2. ábra. md2.cpp által kiszámolt pillanatnyi hőmérsékletek az egyes léptetésekre.

2.feladat

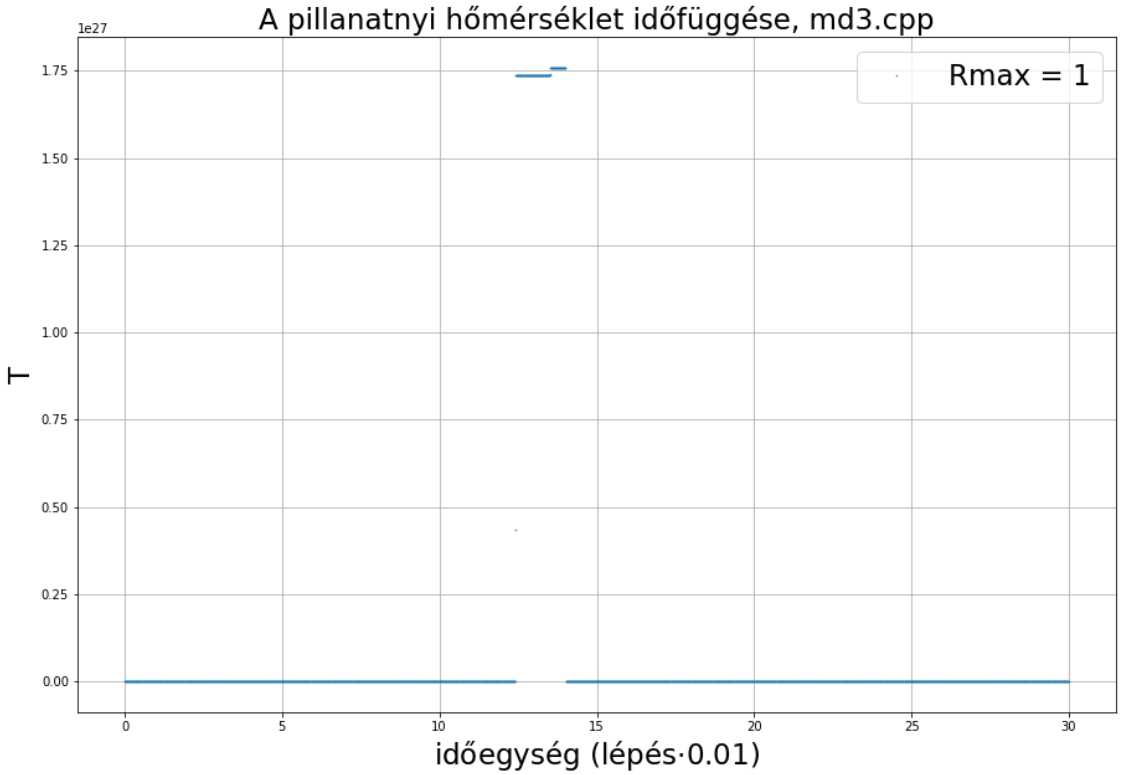
A 2. feladatban az *md3.cpp*-vel kellett ismerkednünk, és játszani az r_{cutoff} , r_{max} és az *updateInterval* paraméterekkel. A 4..3. ábrán láthatjuk, hogy az r_{cutoff} csökkentésével egyre jobban destabilizálódik a hőmérséklet, és ha 1.0 alá csökkentettem, teljesen el is szált (4..5). A 4..4. ábráról azt olvashatjuk le, hogy az r_{max} csökkentésével csökken az egyensúlyi hőmérséklet nagysága, és nő a fluktuációja, és ha 1.5-nél kisebb értékekkel szimuláltam, akkor a fluktuáció már nagyon nagy volt. A 4..6. ábrán pedig az látszik, hogy az *updateInterval* változtatása nem befolyásolja a szimuláció kimenetelét.



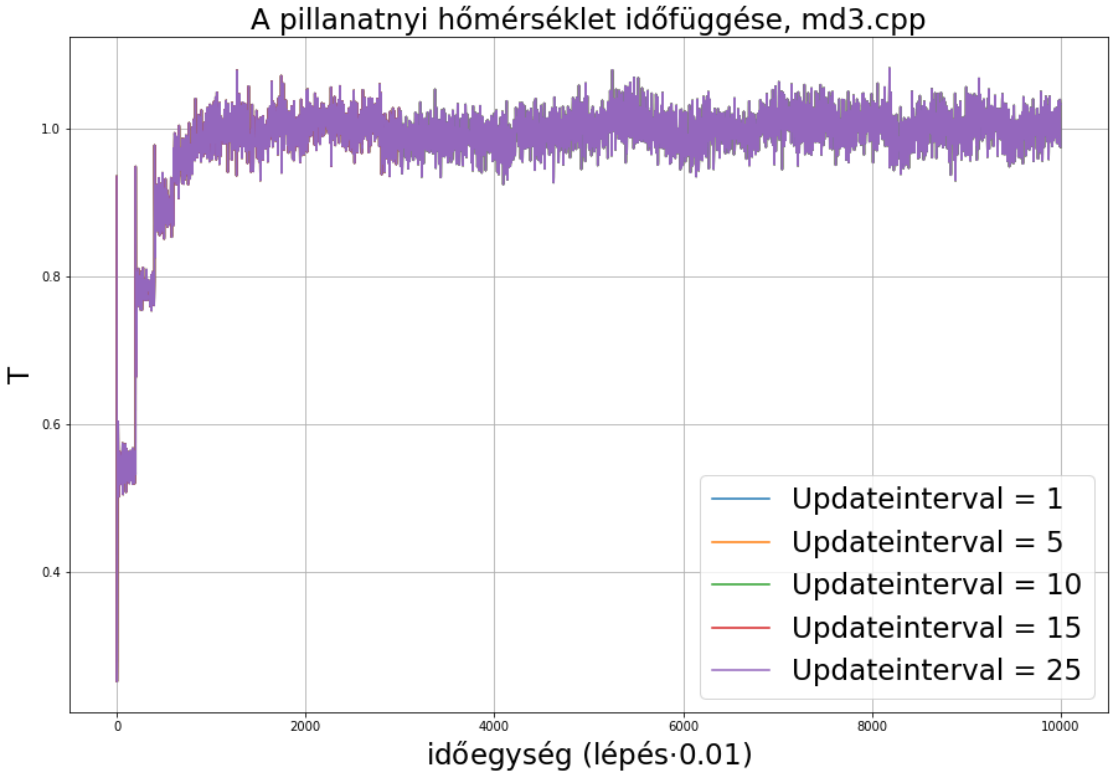
4..3. ábra. md3.cpp által kiszámolt pillanatnyi hőmérsékletek az egyes léptetésekre.



4..4. ábra. md3.cpp által kiszámolt pillanatnyi hőmérsékletek az egyes léptetésekre.



4..5. ábra. md3.cpp által kiszámolt pillanatnyi hőmérsékletek az egyes léptetésekre.



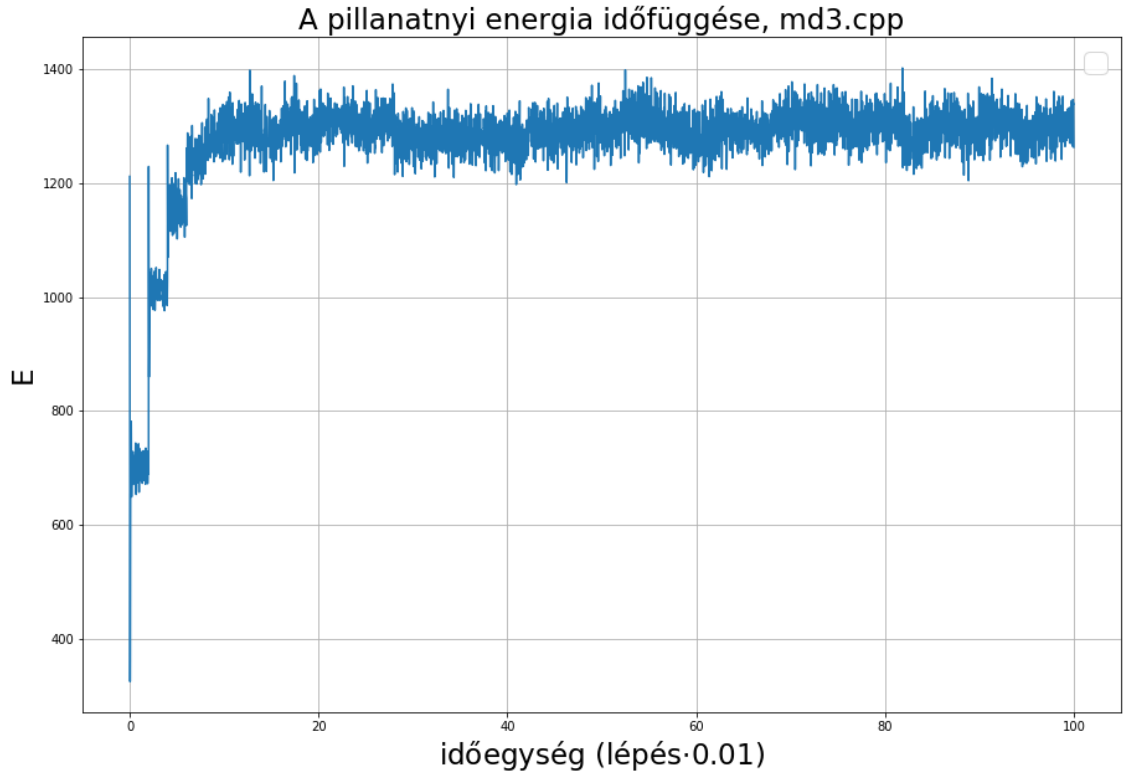
4..6. ábra. md3.cpp által kiszámolt pillanatnyi hőmérsékletek az egyes léptetésekre.

3.feladat

A 3. feladatban ki kellet egészíteni az *md3.cpp* programot úgy, hogy kiszámoljuk a teljes energiát, nyomást, hőkapacitást és a kompresszibilitást.
Amit a 6 programkódrészlet számolt ki.

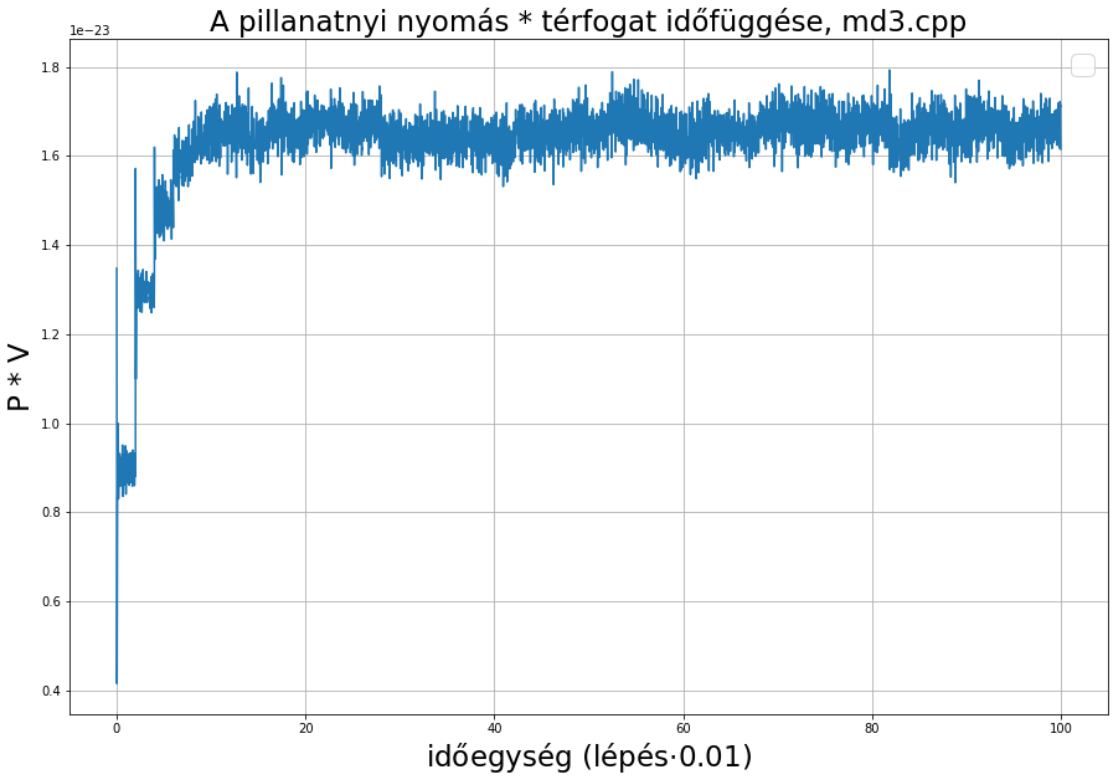
A szimulációt *864* részecskére vizsgáltam, ami az alap beállítás is volt. A kiszámolt termodinamikai mennyiségeket ki is ábrázoltam, ezeket láthatjuk az alábbi (4..7., 4..9., 4..8., 4..10.) ábrákon. Leolvashatjuk róluk, hogy:

- az energia, valamint a nyomás és a térfogat szorzata hasonló görbét rajzol ki, mint a korábban bemutatott hőmérséklet, és ezt is vártuk,
- a hőkapacitás a szimuláció legelején kiugró értéket vesz fel, de aztán nagyon hamar stabilizálódik,
- a kompresszibilitási faktor végig konstans 1 értéket vesz fel*(hiszen ideális gázra történt a szimuláció).*

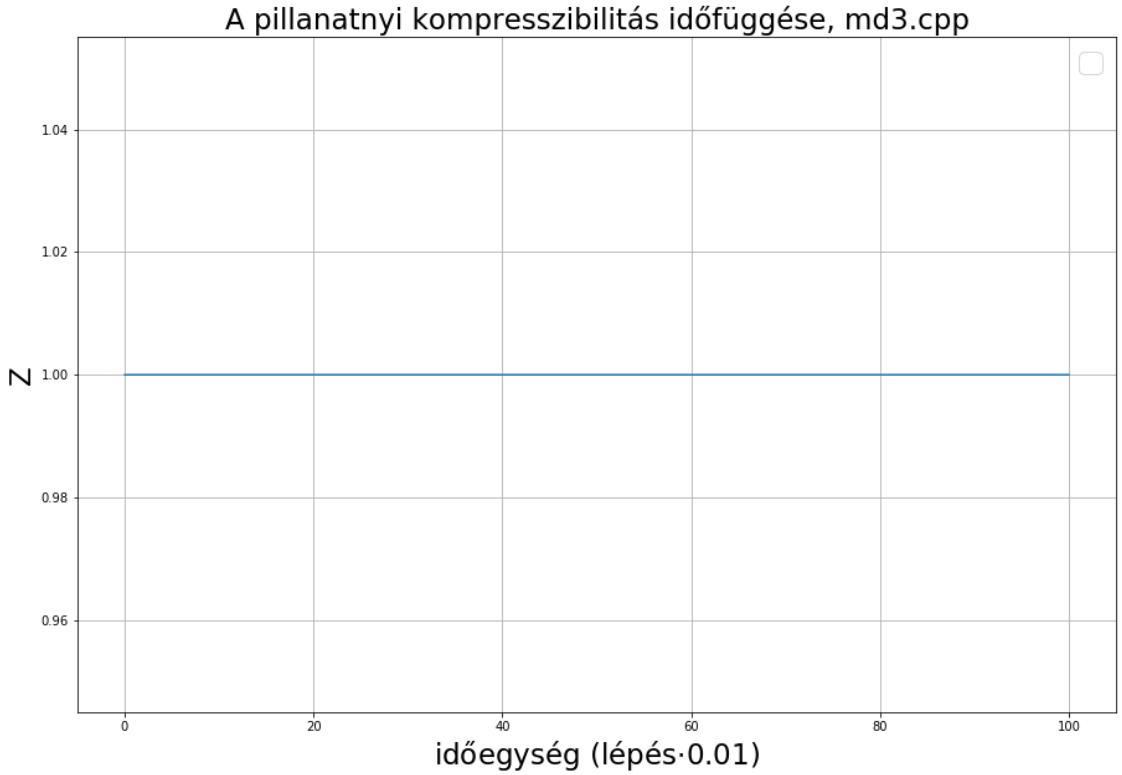


4..7. ábra. md3.cpp által kiszámolt pillanatnyi nyomás megszorozva a térfogattal (Viriál-tétel alapján) az egyes léptetésekre.

4..8. ábra. md3.cpp által kiszámolt pillanatnyi hőkapacitás az egyes léptetésekre.



4..9. ábra. md3.cpp által kiszámolt pillanatnyi nyomás megszorozva a térfogattal (Viriál-tétel alapján) az egyes léptetésekre.



4..10. ábra. md3.cpp által kiszámolt pillanatnyi kompresszibilitás az egyes léptetésekre.

5. Diszkusszió

Jegyzőkönyvemben bemutattam:

- a molekuladinamika szimulációs alapproblémáját, és annak megvalósítását,
- az egyensúlyi hőmérséklet stabilizálódásához szükséges idő függését a numerikus módszertől,
- az egyensúlyi hőmérséklet nagyságának és fluktuációjának függését a Verlet által bevezetett r_cutoff , r_max és az *updateInterval* paraméterektől,
- valamint különböző mérhető termodinamika mennyiségek trendjének időfüggését.

6. Kódrészlet

```
int main(){
        std::vector<double> Energy;
        initialize();
        updatePairList();
        updatePairSeparations();
        computeAccelerations();
        double dt = 0.01;
        ofstream file("T3.data");

        for (int i = 0; i < numOfSimu; ++i){
                Energy_current = 0;
                velocityVerlet(dt);

        double T_instant = instantaneousTemperature();
        file << Energy_current << '\t';
        Energy.push_back(Energy_current);

        double C_v = 0;
        for(int j = 0; j < i; ++j){
                C_v += ((Energy[j] * Energy[j])/i -
                (Energy[j]/i)*(Energy[j]/i));
        }

        C_v *= 1/(boltzmann * T_instant * T_instant);
        file << C_v << '\t';

        double PV = N * boltzmann * T_instant + 1/3 *
        Virial/i;
        file << PV / pow(L, 3) << '\t';

        double Z = PV / (N * boltzmann * T_instant);
        file << Z << '\t';

        file << T_instant << '\n';

        if (i % 200 == 0)
        rescaleVelocities();
        if (i % updateInterval == 0) {
        updatePairList();
        updatePairSeparations();
        }}

        file.close();
}
```