

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 szeretnénk 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 elérhetetlenül 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 vizsgálnunk, hogy azonos kezdőfeltételek mellett mennyi idő alatt relaxál a rendszer, annak egyensúlyi helyzetéhez.

Következő feladatunk a Verlet-féle szomszédosági listával és a távoli potenciálok levágásával operáló korrekciót megvalósító, valamint az enélkül integráló

módszerek futásidejének összehasonlítása volt. Az általam vizsgált karakterisztika a potenciálok levágási határának, valamint a szomszédosági lista frissítési intervallumának módosítására történő futásidőbeli változásokat foglalta magába. Mind a korrekciókkal ellátott, mind pedig az azok nélküli iteratív módszereket teljesen azonos paraméterekkel futtatva, 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 írunk 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 $\mathcal{O}(\tau^4)$ nagyságrendű, a Runge-Kutta $\mathcal{O}(\tau^5)$ hibája mellett, ahol τ a lépéshossz), addig a léptető szabálya miatt - mely a következő:

$$\vec{R}_{n+1} = 2\vec{R}_n - \vec{R}_{n-1} + \tau^2 \vec{A}_n + \mathcal{O}(\tau^4) \quad (1)$$

$$\vec{V}_n = \frac{\vec{R}_{n+1} - \vec{R}_{n-1}}{2\tau} + \mathcal{O}(\tau^2) \quad (2)$$

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 $\mathcal{O}(\tau^2)$ pontosságú. Ezzel ellentétben a Velocity–Verlet-algoritmus \vec{R} -ben és \vec{V} -ben egyaránt csak $\mathcal{O}(\tau^3)$ hibával rendelkezik, de cserébe indítható egy általunk választott kezdőpontból. Léptető szabálya a következő:

$$\vec{R}_{n+1} = \vec{R}_n + \tau \vec{V}_n + \frac{\tau^2}{2} \vec{A}_n + \mathcal{O}(\tau^3) \quad (3)$$

$$\vec{V}_{n+1} = \vec{V}_n + \frac{\tau}{2} (\vec{A}_{n+1} + \vec{A}_n) + \mathcal{O}(\tau^3) \quad (4)$$

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.

A szimuláció 3 különböző módon van megvalósítva, mindegyik mód az előtte levőnek egy bővített verziója. A legelső, **md1** indexű megvalósítás a legalapvetőbb Velocity–Verlet-algoritmust tartalmazza, mely a kezdeti sebességeket random módon választja meg, és a fent tárgyalt léptető szabállyal propagál. Emellett implementálva van itt is, valamint a másik két verzióban is, egy szabályozó alfüggvény is, mely 200 lépésenként ellenőrzi a sebességeket, és ha az azok segítségével, az ekvipartíció tételéből származtatott hőmérséklet nem felel meg a folyamatosan mért pillanatnyi hőmérsékletnek, újraskálázza őket. Ez az a lépés, aminek segítségével a szimuláció erőltetetten propagál az egyensúlyi állapot felé. Ez a mért mennyiségek grafikonján éles törések formájában jelenik meg. Emellett a részecskék mozgásáról készült animáción is drasztikusan szembetűnő az első újraskálázás hatására történő hirtelen sebességcsökkenés[2].

A második, **md2**-vel jelzett szimulációban a kezdősebességeket már a Maxwell–Boltzmann-sebességeloszlás alapján választjuk, mely 3D-ben egyszerűen a Gauss-eloszlás:

$$P(v) = \left(\frac{m}{2\pi k_B T} \right)^{\frac{3}{2}} \cdot e^{-\frac{m(v_x^2 + v_y^2 + v_z^2)}{2k_B T}} \quad (5)$$

Ahogy a gyakorlat leírásában is szerepel[3], a szimuláció a *Numerical Recipes*-ből átvett **gasdev()** függvény segítségével valósítja meg, mely 1 szórású véletlen számokat generál a Box–Müller-algoritmus felhasználásával[4]. Sajnos ez az algoritmus nem tökéletes, a kezdősebességek átlaga - tehát a tömegközéppont sebessége - nem 0. Hogy valóban Gauss-eloszlású sebességeket kapjunk, ahol ez a feltétel igaz, korrigálnunk kell a hibát. Előbb a tömegközéppont sebességét vonjuk le belőlük:

$$\mathbf{v}_i \rightarrow \mathbf{v}_i - \mathbf{v}_{TKP} \quad (6)$$

Majd az így kapott új \mathbf{v}_i sebességeket egy konstans szorzással, létrehozuk a kívánt, inicializáláskor megadott T hőmérsékletet:

$$\mathbf{v}_i \rightarrow \lambda \mathbf{v}_i \quad (7)$$

Ahol a konstans szorzó:

$$\lambda = \sqrt{\frac{2(N-1)k_B T}{\sum_{i=1}^N m \mathbf{v}_i^2}} \quad (8)$$

Ahol az eddigiekben a gyakorlat leírásában is használt formalizmust és jelölést alkalmaztam[3].

A harmadik **md3** indexxel jelölt szimuláció a - majdnem - teljes Velocity–Verlet-algoritmust tartalmazza, melyben már a Verlet által javasolt *szomszédsági lista* is implementálva van, valamint a potenciálok *levágási határa* is értelmezett. Az utóbbi azt a közelítést hozza be a szimulációba, miszerint nagyobb távolságokon a részecskék közti kölcsönhatás elhanyagolhatóan apró. Így ahelyett, hogy a legtávolabbi részecskék közti hatást is kiszámolnánk, minden részecskére csak egy adott **rCutOff** sugarú gömbön belüli hatásokkal foglalkozunk. Hogy ezeket ne kelljen minden alkalommal keresgélni, létrehozunk egy szomszédsági listát. Ez a lista nyilvántartja minden egyes részecskéhez hozzárendelve a tőle **rMax** sugarú gömbön belül található összes többi részecskét. Ilyen esetben ha ki akarjuk számítani egy lépésben az egyik részecskére ható erőket, akkor egyszerűen csak a szomszédsági listán kell az iterációt elvégezniünk a rendszerben található összes molekula helyett.

Ezt a listát **updateInterval** lépésenként frissítjük. Verlet másik közelítése alapján azért nem kell ezt a frissítést minden lépésben megtenniünk, mert a részecskék sebessége véges. Ha megfelelően választjuk meg **rMax** és **rCutOff** értékeit, akkor **updateInterval** lépésen belül azok helyzete nem változik annyira meg, hogy ez bármennyire is befolyással legyen a szimuláció pontosságára. Verlet javaslata alapján a fent tárgyalt paraméterek értékeit a következőképp kell megválasztanunk:

$$\mathbf{rCutOff} = 2.5r_0$$

$$\mathbf{rMax} = 3.2r_0$$

$$\mathbf{updateInterval} = 10$$

Az ezekben megjelenő r_0 jelentését és értékét a következő két részben tárgyalom.

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[5] írja le:

$$V(r) = 4V_0 \left[\left(\frac{r_0}{r} \right)^{12} - \left(\frac{r_0}{r} \right)^6 \right] \quad (9)$$

Ennek alakja könnyen megmagyarázható. 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 gradienseiből számíthatjuk legegyszerűbben:

$$\vec{F}(\mathbf{r}) = -\nabla V(r) = \frac{24V_0}{r^2} \left[2 \left(\frac{r_0}{r} \right)^{12} - \left(\frac{r_0}{r} \right)^6 \right] \mathbf{r} \quad (10)$$

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:

$$V_0 = r_0 = m = 1 \quad (11)$$

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:

$$\mathcal{H} \equiv E = \frac{m}{2} \sum_{i=1}^N \mathbf{v}_i^2 + \sum_{i \neq j} V(|\mathbf{r}_i - \mathbf{r}_j|) \quad (12)$$

Ahol $V(|\mathbf{r}_i - \mathbf{r}_j|)$ az i és j indexű részecskék közti Lennard-Jones-potenciál, \mathbf{v}_i pedig az i -edik részecske sebessége.

Másodikként a moláris, konstans térfogaton mért hőkapacitást határoztam meg, melyet a gáz fundamentális egyenletéből, vagy a fluktuáció-disszipáció tételéből kaphatunk[6] a következő módon:

$$C_V = \left(\frac{\partial E}{\partial T} \right)_V = \frac{1}{k_B T^2} \left[\langle E^2 \rangle - \langle E \rangle^2 \right] \quad (13)$$

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[7]. Az

$$\left[\langle E^2 \rangle - \langle E \rangle^2 \right] \quad (14)$$

szórásban szereplő átlagokat időátlagokként közelítettem, ahogy az a feladat kiírásában is javasolva volt[3].

Végezetül a nyomást és a kompresszibilitási faktort a Viriál-tétel segítségével mértem, melyből felírható az alábbi összefüggés:

$$PV = Nk_B T + \frac{1}{3} \left\langle \sum_{i < j} \mathbf{r}_{ij} \cdot \mathbf{F}_{ij} \right\rangle \quad (15)$$

Melyből a kompresszibilitási faktor:

$$Z = \frac{PV}{Nk_B T} \quad (16)$$

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 funkcionalitását, 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-eket 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[8].

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 könyvtárakat 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[8].

5. KIÉRTÉKELÉS

Minden feladatot az (1)-es pontban is említetteknek megfelelően zárt, keményfalú rendszerekben vizsgáltam, amiről a részecskék tökéletesen rugalmas ütközéssel pattantak le. Minden futtatás során 64 db részecske mozgását szimuláltam, az md2 és md3 esetében rho = 0.95 sűrűségértékkel és T = 1.0 egyensúlyi hőmérséklettel. Az md3 programban a Verlet-féle gyorsításokhoz a (3.1)-ben ismertetett, Verlet által kavaszt paramétereket használtam. Első feladatként a különböző szimulációk közti különbségeket kellett feltárnunk, azon belül is a rendszer relaxációs idejét vizsgáltuk. Az (1)-es ábrán találhatóak az egyes md programok alapértelmezett

kimenetei, a pillanatnyi hőmérséklet értékek. A sebességkorrekció okozta - már említett - levágás egyértelműen látszódik minden 200. lépés esetén. Az első - legdrasztikusabb - ilyen levágás az (1) - (7) ábrák mindegyikén szaggatott, függőleges zöld vonallal jelölve is van. Míg az md1 esetében csak az első ilyen levágás látható ránézésre, addig az md2 esetén a 2. is, az md3-nál pedig még a 7. ilyen korrekciónál is látható egy éles törés. Mikor ez a karakterisztika a zajban már elveszik, akkor mondhatjuk, hogy a szimulált rendszer elérte az egyensúlyi helyzetét és sztochasztikus rendszer módjára akörül oscillál. Az md1 esetében az előre beadott $T = 1.0$ K hőmérsékletet már az első levágás után eléri rendszer, míg a többinél - ahogy említettem - jóval később. Ez betudható annak a ténynek, hogy a két másik rendszer az elején elszálló viselkedést mutat, a Maxwell-Boltzmann-eloszlással választott sebességek jóval nagyobbak, mint a random módon választott egyensúlyi helyzet körüliek. Emiatt a rendszer összenergiája az elején még jóval nagyobb, és az újraszámítás csak több lépésben tudja azt az egyensúlyba propagálni. Megemlítendő, hogy az md2 és az md3 jelű szimulációk a bemeneti ρ paraméter változtatására - mely az egységterfogatban található kezdeti részecskesűrűséget szabályozza - nagyon érzékenyek. $\rho > 1$ esetén rendkívüli sebességgel szállnak el, $\rho = 1.05$ esetén az md3 esetében elérve a 10^{24} nagyságrendet is. A jelentős kezdeti sebességkülönbséget az egyes szimulációkról készült animációk is mutatják[2].

Második feladatunk a futásidő vizsgálata volt, melyet a (6)-os - következő - részben tárgyaltam részletesebben, minden azzal kapcsolatos információ abban található.

A rá következő feladat különböző mennyiségek vizsgálatát foglalta magába, melyek elméleti megfontolásait a (3.3)-as részben tisztáztam. Azok alapján készítettem el a (2) - (7) ábrákat, melyeken sorrendben a rendszer teljes energiája (2), az energia időátlaga ($\langle E \rangle$) (3), annak oszcillációja ($\langle E^2 \rangle - \langle E \rangle^2$) ($\langle E \rangle$) (4), a moláris hőkapacitás (5), a nyomás (6), valamint a kompresszibilitási faktor (7) látható, mindhárom szimuláció futtatása során.

A teljes energia egy zárt rendszerben meg kell maradjon. Ez bizonyos értelemben itt is így van, ami a (2)-es ábra grafikonjain is látható. Mikor a szimuláció 200 lépésenként korrigál, a teljes energia vele együtt változik, azonban ezen korrigációk között minden esetben állandó, fluktuációk nélkül. Ez alól egyedüli kivétel az md2 és md3 szimuláció, első korrigáció előtti viselkedése. Az utóbbi esetében az energia változás nagyon jelentős, míg az elsőben az csak apró fluktuáció. Végül mindkettő szimuláció beáll viszonylagosan a -100 értékre. Az md3 láthatóan azért pontosabban, mint az md2, míg az md1 közelében sincs a másik kettőnek. Még csak nem is egy konkrét érték körül fluktuál, hanem folyamatosan csökken.

A kezdeti sikerek után a moláris hőkapacitás értékére egy meglepő, előre nem várt értéket kaptam, mely szemmel láthatóan abnormálisan nagy, mégis minden esetben mintha tartana egy konkrét értékhez, melyek a (5)-es ábra grafikonjairól leolvashatóak. Ezt az értéket a fluktuáció-disszipáció elve alapján számoltam, melyet a (3.3)-as fejezetben ismerttettem. Ez erősen függ a $\langle E^2 \rangle - \langle E \rangle^2$ ($\langle E \rangle$)

fluktuációtól, melynek értékeit a (4)-es ábrán vázoltam fel. Ez el van osztva a Boltzmann-állandó egy - nem túl nagy - konstansszorosával, mely így végérvényben egy roppant nagy értéket fog eredményül adni. A pontos okot, amiért az energiák itt így elcsúsznak, nem tudtam a jegyzőkönyv beadásáig megtalálni.

A rendszerben uralkodó nyomást szintén a (3.3) részben ismertett módon mértem. A szimuláció minden léptetésénél kiszámoltam a $\sum_{i < j} \mathbf{r}_{ij} \cdot \mathbf{F}_{ij}$ átlagát, melyből megkaptam PV értéket. Miután V ismert volt, a nyomást egyszerűen a kettő osztásával kaptam meg. Ennek nagyon apró nagysága a (6)-os ábrán látható.

A fenti PV értékből végül könnyen ki tudtam számítani a kompresszibilitási faktort, mely meglepően egy konstans $Z = 1$ értéket mutatott, ahogy az egy ideális gáz esetében várható. Ezek alapján a szimulált rendszerek minden esetben ideális gázokként viselkedtek. Előzetes feltevés volt, hogy a részecskeszám növelésével ez az eredmény változik, azonban az általam végigpásztázott - még elfogadható futásidővel rendelkező - mérettartományban ilyen nem észleltem. Ez betudható a nyomás roppant apró értékének, mely döntős súllyal szerepel az egyenletben. Ténylegesen csak az Avogadro-szám nagyságrendjében találhatnánk esetleg változást (hisz a P nyomás értéke is ilyen nagyságrendű).

6. FUTÁSIDŐ

Az md3 indexxel jelzett programban több bemeneti paramétert változtatva térképeztem fel a szimuláció futásidejét. Ezek a paraméterek az rCutOff a - (3.1)-es részben tárgyalt - potenciálok levágási távolsága, rMax a szomszédsági lista nyilvántartási távolsága, valamint az updateInterval, a szomszédsági lista frissítései közt eltelt intervallum voltak. Ezek során a vizsgált intervallumokon belül minden kombinációt végigpróbáltam, ez összesen végül 2310 variációt jelentett. Egymásba ágyazott ciklusok segítségével a lent felsorolt intervallumokat vizsgáltam meg.

A levágási távolság esetén:

$$\text{rCutOff} \in [2.5, 3.5] \quad (17)$$

Ahol a léptetés 0.1-enként volt. Itt és rMax esetén is az értékek (3.2) alapján $r_0 = 1$ mértékben vannak felírva. A szomszédsági lista nyilvántartási távolságának határa esetén:

$$\text{rMax} \in [\text{rCutOff} + 0.0, \text{rCutOff} + 2.5] \quad (18)$$

Ahol a lépésköz az előzőhöz hasonlóan, szintén 0.1 volt. Az itt megjelenő rCutOff mennyiség azt jelentette, hogy értéke az rCutOff függvényében változott, annál sosem volt kisebb. Míg végül a szomszédsági lista frissítései között eltelt lépések száma a következők voltak:

$$\text{updateInterval} \in [5, 15] \quad (19)$$

Ahol updateInterval minden esetben egész, a ciklusok közti lépésköz tehát 1 volt.

Ezen fentiek segítségével a 3 változó függvényében meg tudtam határozni a futásidőket. Mivel ezek ábrázolásához egy 4 dimenziós függvényt kéne felrajzolnunk - amit sajnos nem lehet - ezért csak annak projekciót tudjuk megjeleníteni. Ezek a (8)-os ábrán olvashatóak, melyeken látható 3 grafikon

minden különböző párosításban tartalmaz két-két változó mennyiséget és azok függvényében ábrázolt futásidőt. Az utána következő ábrák (9) - (11) ezen 3D grafikonok még további 2D projekciót tartalmazza. Róluk az olvasható le, hogy Verlet, (3.1)-ben olvasható javaslata viszonylag helytálló, valóban alacsony értékek tartoznak a javasolt értékekhez, de jelen adatsor esetén több helyen is voltak jobb futásidők. Ilyen pl. az `rCutOff = 2.8`, valamint az `rMax = 3.1` feltételek.

Az `md2` programmal összehasonlítva a futásidő el-

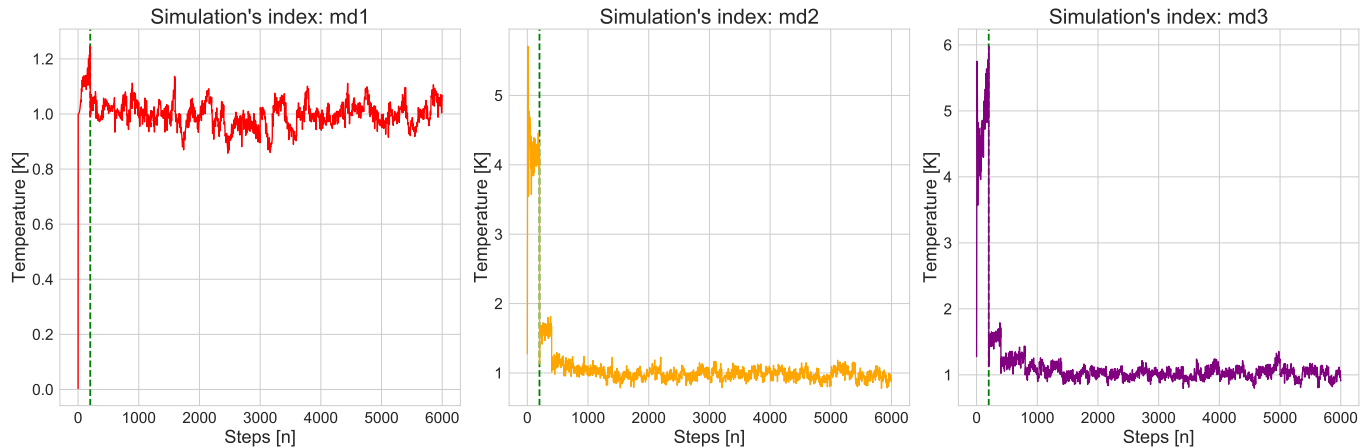
lenben mindig sokkal jobb volt, egyik esetben sement 1 s fölé, míg az `md2` átlagideje 1.3 s körül volt. Ez az ábrán nem szerepel, azonban a `difference.py` program futtatása során létrejövő kimeneti file első sora tartalmazza ezt az átlagos értéket az összes többi `md3` futásidővel egyetemben. Ezek mindegyike elérhető azonban GitHubon[8].

7. DISZKUSSZIÓ

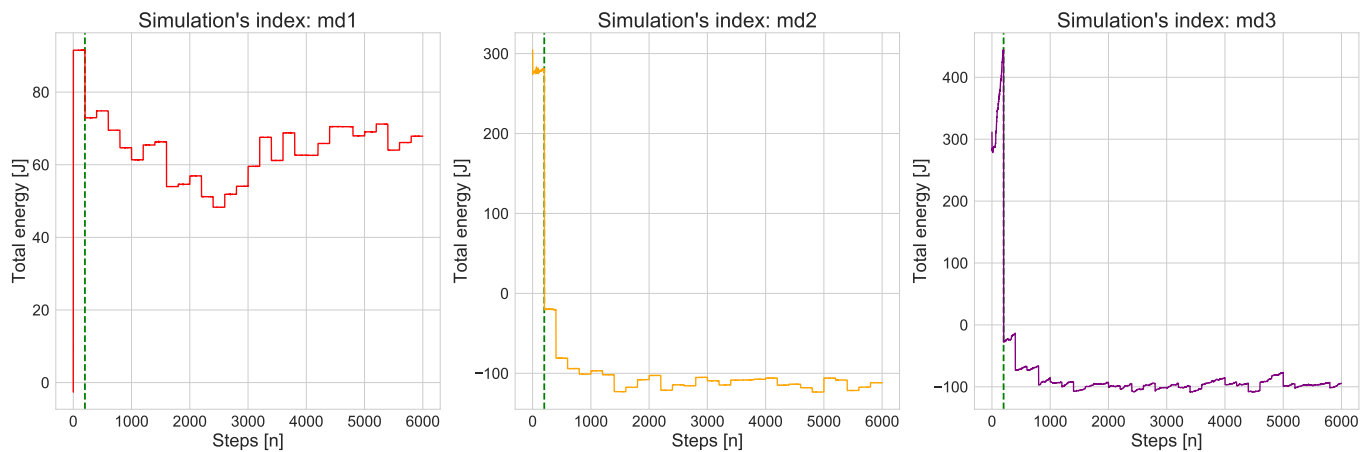
-
- [1] Loup Verlet. “Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules”. In: *Physical review* 159.1 (1967), p. 98.
 - [2] Pál Balázs’s Channel — YouTube. [Online; opened at March 23, 2019]. 2019. URL: <https://www.youtube.com/channel/UCBDSB7PdQ3E919WSBsTy7cQ>.
 - [3] József Stéger, István Csabai. *Számítógépes szimulációk – Molekuladinamika*. [Online; opened at March 25, 2019]. 2019. URL: <https://stegerjosef.web.elte.hu/teaching/szamszim/moldin.pdf>.
 - [4] William H Press et al. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.
 - [5] 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](https://doi.org/10.1098/rspa.1924.0082).
 - [6] Igor Vilfan. *Lecture Notes in Statistical Mechanics – Foundations*. [Online; opened at March 26, 2019]. 2000. URL: <http://www-f1.ijs.si/~vilfan/SM/ln1.pdf>.
 - [7] Stephen T. Thornton and Andrew Rex. *Modern physics for scientists and engineers*. Cengage Learning, 2012.
 - [8] Pál Balázs. *ELTE Computer Simulations 2019 — GitHub*. [Online; opened at March 23, 2019]. 2019. URL: https://github.com/masterdesky/ELTE_Comp_Simulations_2019.

APPENDIX A

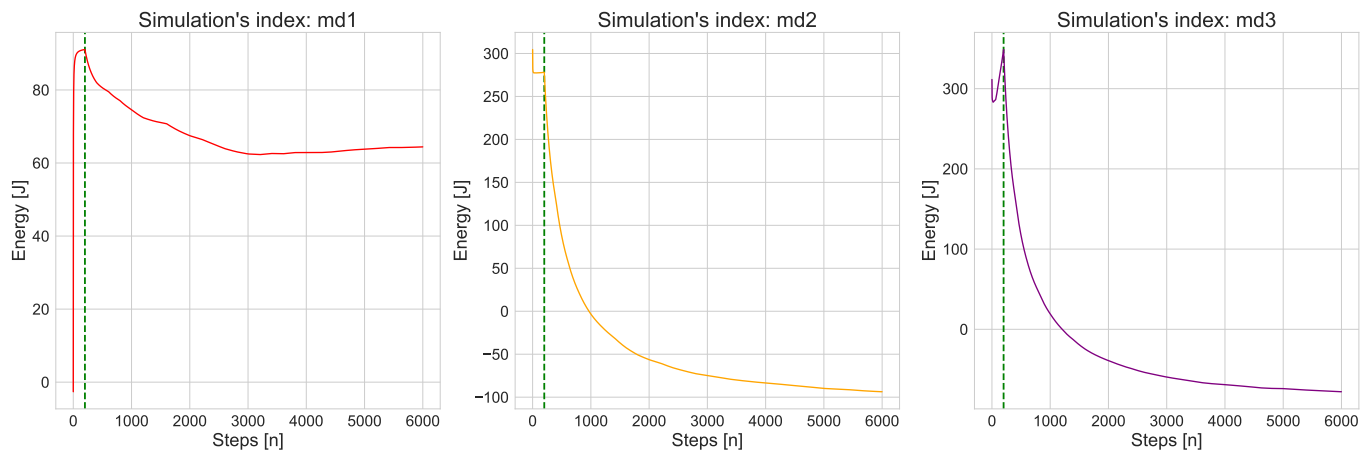
A.1.1 MÉRENDŐ MENNYISÉGEK ADATAI



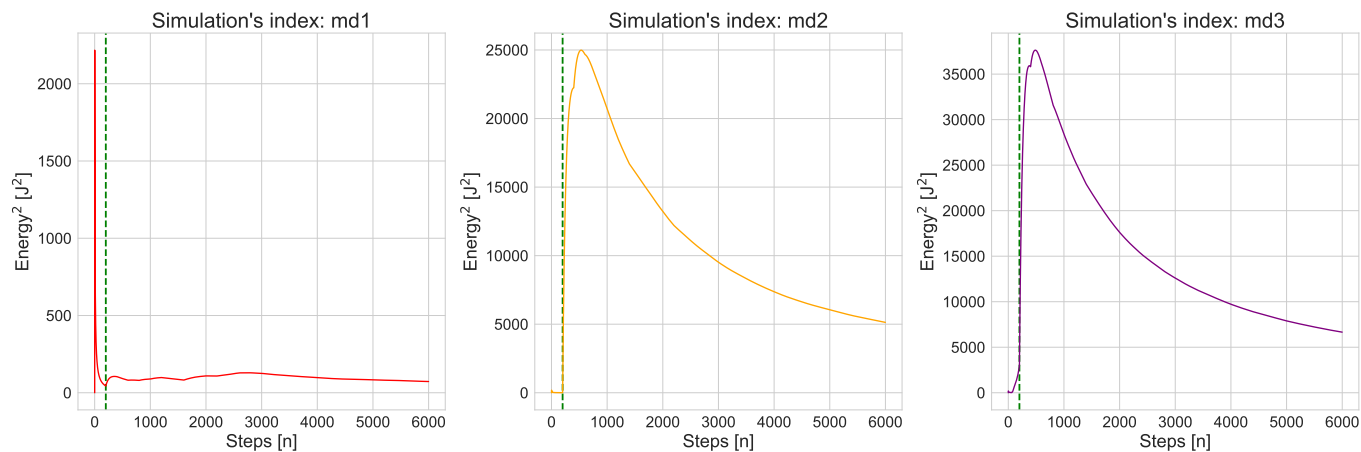
1. ábra. Pillanatnyi hőmérsékletek zárt rendszerben, $N = 64$ részecske esetén



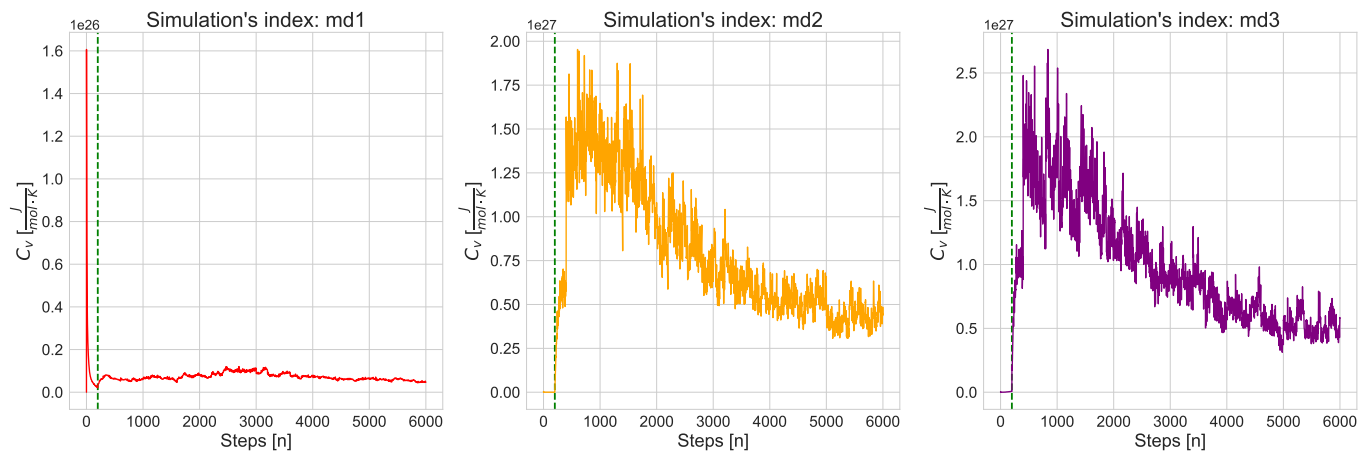
2. ábra. A zárt rendszer teljes energiája $N = 64$ részecske esetén



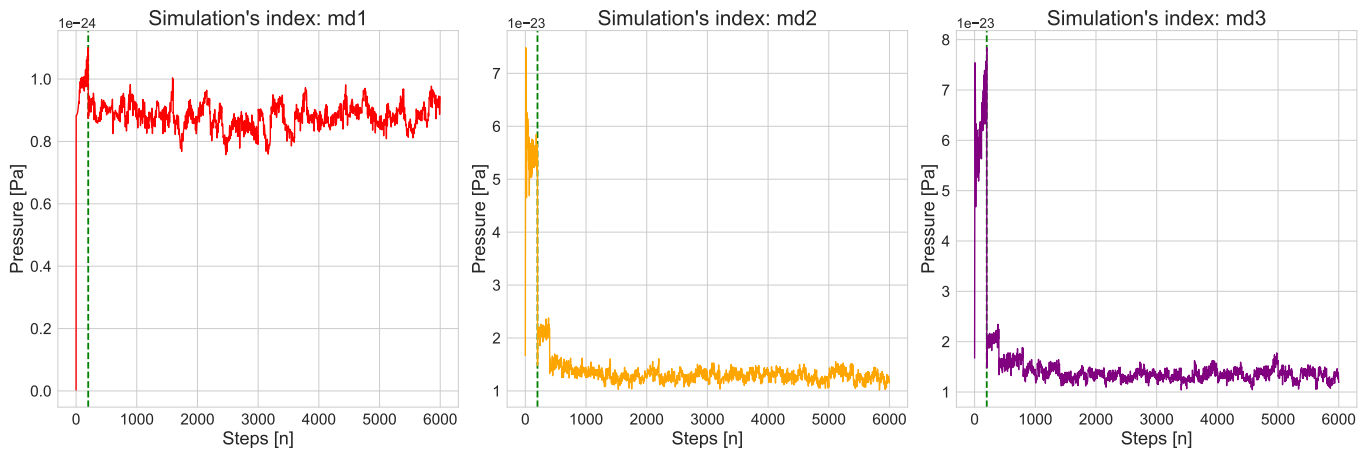
3. ábra. A mérhető energia időátlaga ($\langle E \rangle$) $N = 64$ részecske esetén



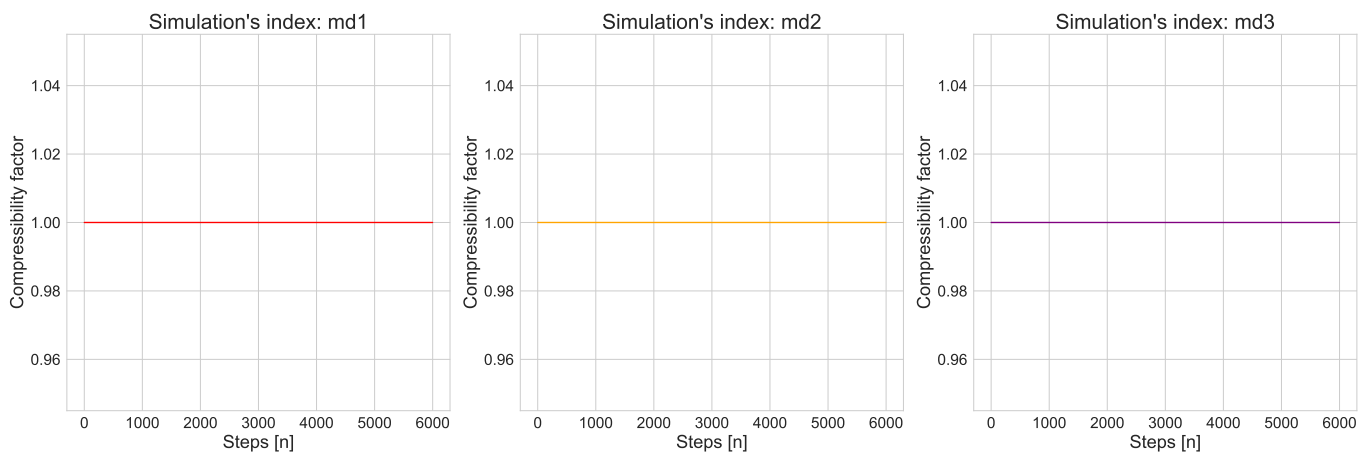
4. ábra. Az energia oszcillációja ($\langle E^2 \rangle - \langle E \rangle^2$) $N = 64$ részecske esetén



5. ábra. A zárt rendszerben mért moláris hőkapacitás $N = 64$ részecske esetén

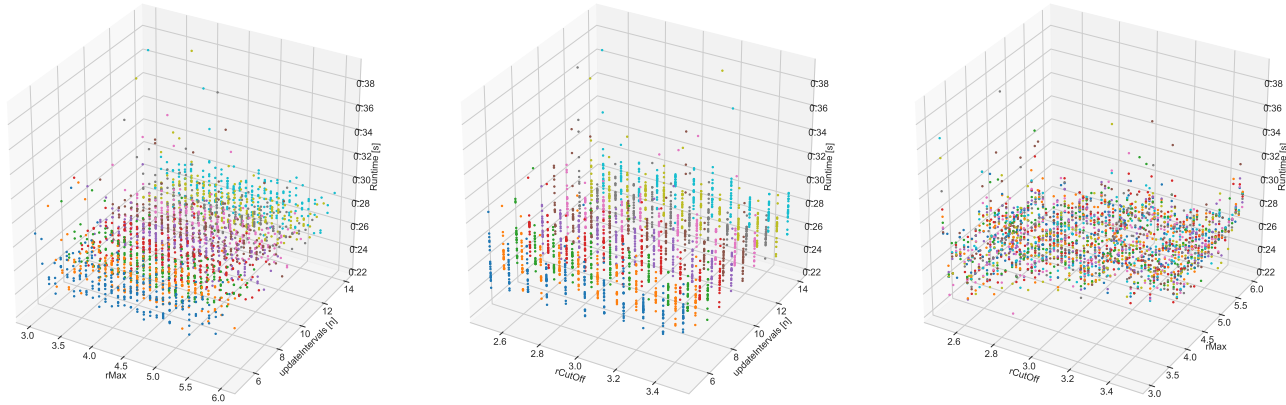


6. ábra. A zárt rendszer nyomása $N = 64$ részecske esetén

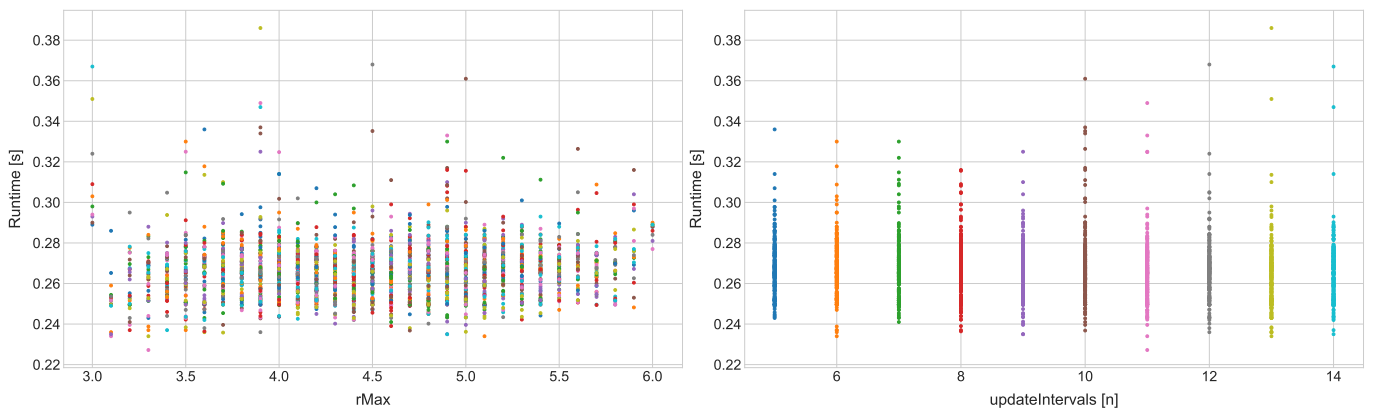


7. ábra. A zárt rendszer kompressziós együtthatója $N = 64$ részecske esetén

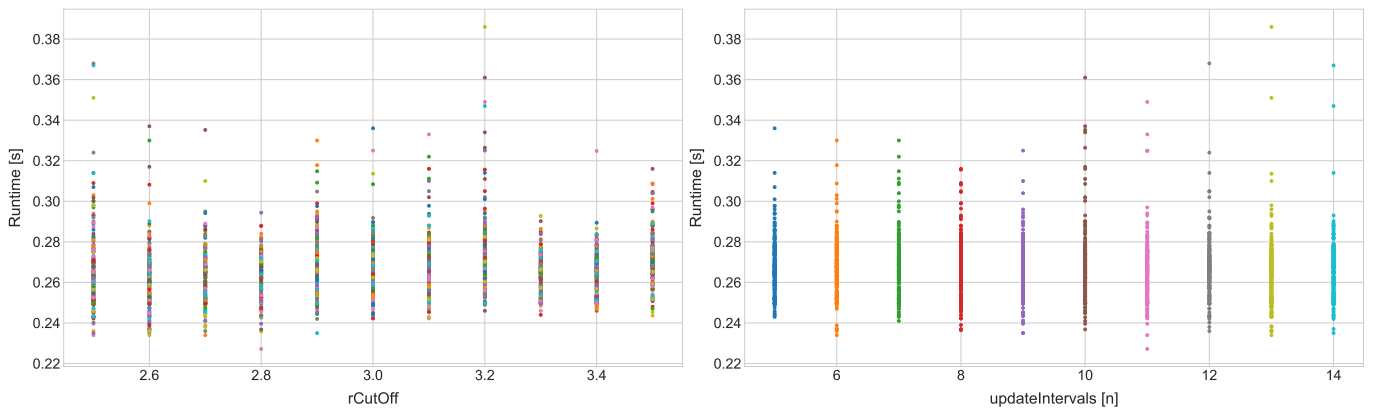
A.1.2 FUTÁSIDŐK



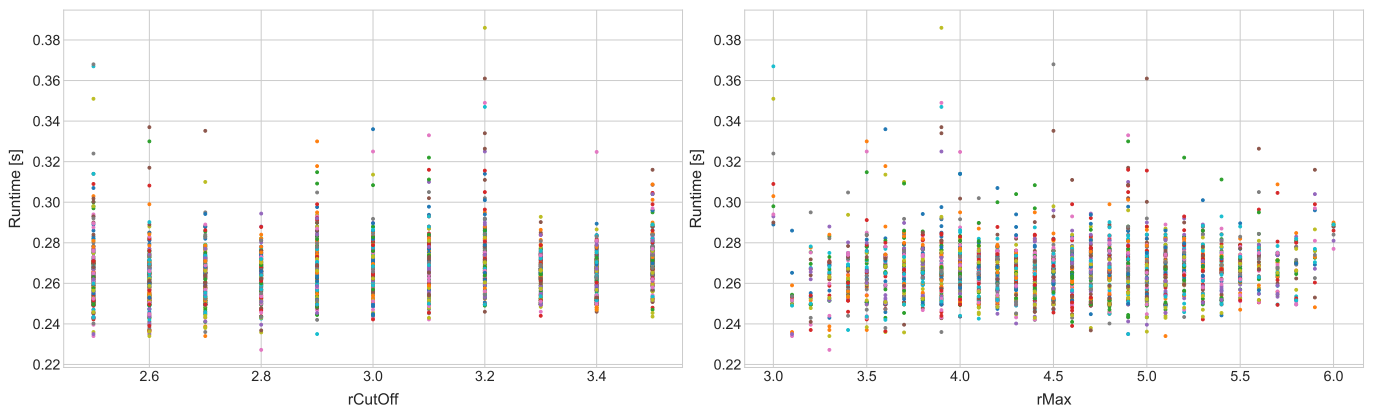
8. ábra. A futásidők teljes 4D terének 3D projekciói. Az azonos szimulációk mindhárom képen azonos színekkel vannak jelölve.
 Bal szélső ábra: Az egyes szimulációk futásideje az `rMax` és az `updateInterval` függvényében
 Középső ábra: A futásidők az `rCutOff` és `updateInterval` függvényében
 Jobb szélső ábra: A futásidők az `rCutOff` és `rMax` függvényében



9. ábra. A futásidők első további 2D projekciói. Itt azok az `rMax` és `updateInterval` paraméterek függvényében vannak ábrázolva.



10. ábra. A futásidők második további 2D projekciói. Itt azok az `rCutOff` és `updateInterval` paraméterek függvényében vannak ábrázolva.



11. ábra. A futásidők harmadik további 2D projekciói. Itt azok az `rCutOff` és `rMax` paraméterek függvényében vannak ábrázolva.