Molekylær Dynamikk

Oblig 3

av

Øyvind Sigmundson Schøyen
Kandidatnummer: 30

Avsluttande prosjekt i FYS3150

FYS3150 Computational Physics
Det matematisk-naturvitskaplege fakultet
Universitetet i Oslo

Desember 2014

Samandrag

I dette prosjektet har me tatt for oss modellering av argon på atomært nivå. Me har vore interesserte i å lage eit program som skal lage ein atomstruktur etter eige ynskje. I tillegg har me ville sjå på korleis eit slikt system utvikler seg over tid og måle statistiske eigenskapar ved det. For å modellere eit så realistisk resulatat innenfor det som er mogleg å køyre på ein pc har me utnytta cellelister for å auke hastigheita på programmet. For krafta har me nytta Lennard Jones potensialet og som integrator har me brukt Velocity Verlet-algoritma. Programma våre er objektorienter `C++`-kode med eit `Python` rammeverk som skal enklast mogleg køyre programmet vårt for forskjellige parametrar og plotte verdiar. Me nyttar VMD for å visualisere atoma i rommet. All kjeldekode ligg på github.

Rett litt på denne...

`https://github.com/Schoyen/molecular-dynamics-fys3150`

Innhald

Introduksjon

Oppgåva me er gjevne har gått ut på å modellere eit fysisk system samt bruke objektorientert programmering til å holde ein ryddig, oversiktlig, samt effektiv kode. I byrjinga er me gjeve ein kode som lagar 100 argon atom og gjer dei ein tilfeldig retning og hastighet. Gjeve lang nok tid vil atoma drive vekk. Me vil difor nytte "periodiske randbetingelsar" for å halde atoma i nærleiken. Grunna hastigheitar gjevne frå Maxwell-Bolzmann distribusjon vil systemet ha ein ikkje-null rørslemengde som me vil fjerne. Me vil deretter lage ein krystallstruktur kor me startar simuleringa av atoma. I eit slik lukka system vil total energien vere bevart, men grunna numerisk avrunding er det ikkje alltid dette held mål. Ein stabil alogritme som me vil nytte er "Velocity Verlet" som er ein symplektisk integrator. Me vil no byrje å måle statistiske eigenskapar som energi og temperatur. For å kunne køyre koden for store system vil me derimot utvikle ein kjappar algoritme når me rekner ut krafta mellom atompara. Dette løyser me med cellelister. Til slutt, i fyrste del av prosjektet, legg me til ein termostat som let oss kontrollere temperaturen i systemet.

God lesning!

Fysikken bak molekylær dynamikken

I denne delen av rapporten vil me sjå på dei forskjellige eigenskapane me måler i MD-koda vår. Eventuelle måleresultat vil bli vist i resultat-seksjonen.

Oppsett

Systemet med atom me set opp krev nokre tilpassningar for å gje eit realistisk resultat.

Face-Centered Cubic Lattice

Det fyrste me vil gjere å plassera atom i ein krystallstruktur. For Argon vil me då nytte "face-centered cubic lattice" (FCC). Me plasserer fire og fire atom i ei einingscelle. Posisjonane til kvart atom vil vere gjeve ved

$$\begin{aligned}
\mathbf{r}_1 &= 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}, \\
\mathbf{r}_2 &= \frac{b}{2}\mathbf{i} + \frac{b}{2}\mathbf{j} + 0\mathbf{k}, \\
\mathbf{r}_3 &= 0\mathbf{i} + \frac{b}{2}\mathbf{j} + \frac{b}{2}\mathbf{k}, \\
\mathbf{r}_4 &= \frac{b}{2}\mathbf{i} + 0\mathbf{j} + \frac{b}{2}\mathbf{k}.
\end{aligned}$$

Her vil b vere ein konstant, me skal seinare sjå på korleis trykket er avhengig av denne.

Maxwell-Boltzmann og fartsmoment

Etter at atoma vert plasserte i einingscellene vil me gje dei ein liten starthastighet kor me nyttar Maxwell-Boltzmann distribusjon. Då vil

$$\mathbf{v} \propto \sqrt{T}.$$

Hastigheita til atoma vil bli fordelte tilfeldig i rommet. Resultatet er at systemet har eit ikkje-null fartsmoment. Før me byrjar å rekne ut nye posisjonar vil me fjerne dette fartsmomentet. Me vil då finne den totale hastigheita til systemet og trekke denne frå kvart atom.

$$\begin{aligned}
\mathbf{V} &= \frac{1}{M}\sum_i^N m_i \mathbf{v}_i^{\text{før}}, \\
\mathbf{v}_i^{\text{etter}} &= \mathbf{v}_i^{\text{før}} - \mathbf{V}, \qquad i \in [1, N],
\end{aligned}$$

kor $\mathbf{V}$ er den total hastigheita til systemet, N er antal atom, M er den totale massa til alle atoma summert opp. Dette bidrar til at systemet vårt ikkje har ein total hastigheit, men heller står i ro.

Lennard Jones

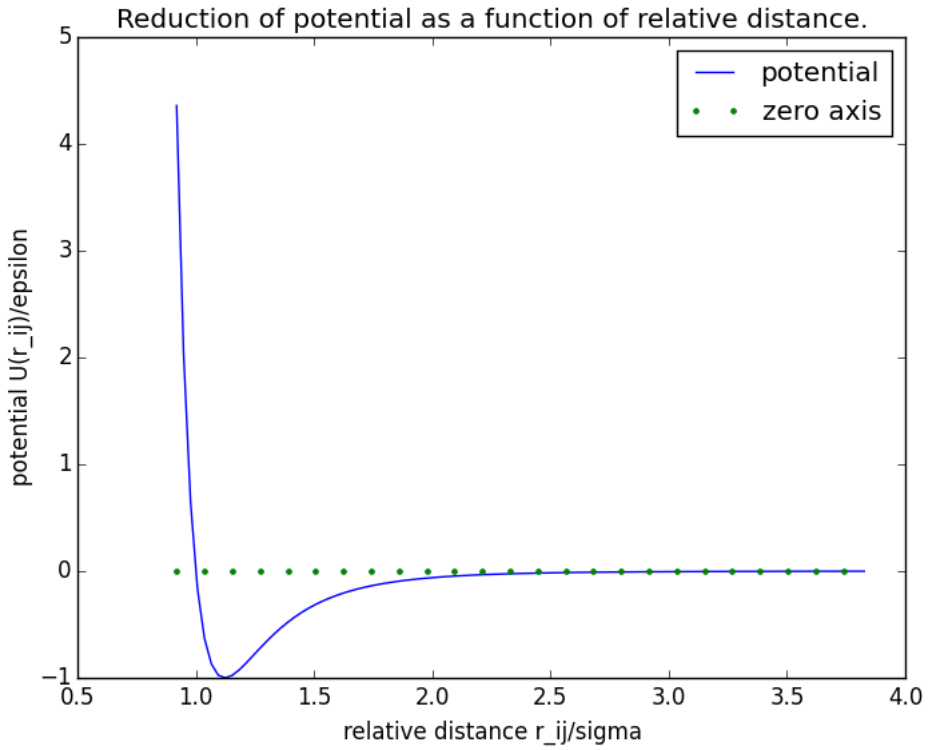
For å få ein fysisk effekt på systemet må me byrje å rekne ut krafta mellom atoma. Me nyttar Lennard Jones potensialet for å finne krafta mellom atompara. Lennard Jones har den eigenskapen at potensialet tek med den fråstøytande og den tiltrekkande krafta mellom para. Potensialet er gjeve ved

$$U(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right],$$

kor r_{ij} er avstanden frå atom i og j, σ og ϵ er konstantar som bestemmer kva distanse potensialet er null og kor djup potensialbrønnen skal vere. Me finn krafta ved å ta den negative gradienten til potensialet. Då får me

$$\begin{aligned} \mathbf{F}(\mathbf{r}_{ij}) &= -\nabla U(r_{ij}) = -\frac{\partial U(r_{ij})}{\partial r_{ij}} \\ &= -4\epsilon \left[12 \left(\frac{\sigma^{12}}{r_{ij}^{14}} \right) - 6 \left(\frac{\sigma^{6}}{r_{ij}^{8}} \right) \right] \mathbf{r_{ij}}. \end{aligned}$$

Grunna dei høge potensane kan me sjå at denne krafta vil kjapt gå mot null. Denne eigenskapen gjer at me kan effektivisere koda vår ved berre å rekne ut krafta mellom atom kor avstanden mellom dei er innenfor ein radius r_{cut}. Eit plott over U/ϵ mot r_{ij}/σ vil gje oss ein indikator kva tid krafta vil vere null.



Figur 1: I plottet kan me lett sjå kor dei tiltrekkjande og kor dei fråstøytande kreftene har sine styrker. Då krafta er gjeve som den negative gradienten til potensialet vil me kunne sjå at i det atoma kjem veldig nære einannan vil det virke ei kraftig fråstøyting (r_{ij}^{12}-delen) medan det for større avstand vil virke ei svakare, men med større rekkevidde, tiltrekkande kraft (r_{ij}^{6}-delen).

Frå figuren les me av

$$\frac{r_{\text{cut}}}{\sigma} \approx 2.5 \qquad \Rightarrow \qquad r_{\text{cut}} \approx 2.5\sigma.$$

Utanfor denne radiusen vil me ikkje rekne krafta mellom atompara.

Statisiske målingar

Hensikten med koda vår er for å måle eigenskapar i eit system med atomar.

Energi

Det enklaste å måle vil vere energien i systemet. Me har allereie implementert eit uttrykk for den potensielle energien med Lennard Jones. Då finner me den kinetiske energien ved formelen

$$E_k = \sum_i^N \frac{1}{2} m_i v_i^2,$$

kor N er antal atomar i systemet. Total energien vil då vere gjeve ved

$$E = E_k + U.$$

For eit system av atom med dei randbetingelsane me har satt forventer me energibevaring. Grunna numerisk avrunding vil det alltid oppstå små fluktuasjonar, men desse forventer knapt skal vere synlege.

Då me ikkje reknar krafta mellom atompar utanfor r_{cut} må me endre litt på Lennard Jones potensialet slik at me har energibevaring. Potensialet vil nå bli skalert etter likninga

$$U_{\text{skalert}}(r_{ij}) = \begin{cases} U(r_{ij}) - U(r_{\text{cut}}), & r \leq r_{\text{cut}} \\ 0, & r > r_{\text{cut}}. \end{cases}$$

Temperatur

Temperaturen speler ei stor rolle i systemet vårt. Me er interesserte i å finne ut kva tid strukturen smelter og kor tid ho held seg stabil. Me finn ho ved ekvipartisjonsteoremet som seier

$$\langle E_k \rangle = \frac{3}{2} N k_B T \qquad \Rightarrow \qquad T = \frac{2}{3} \frac{E_k}{N k_B},$$

kor T då angjer den momentane temperaturen då me ikkje tek gjennomsnittet av E_k. Temperaturen vil fluktuere då den kinetiske energien vil endre seg for kvart tidssteg. Etterkvart vil me nå eit stabilt likevektspunkt kor temperaturen held seg nokonlunde jamnt.

Tettleik

Me vil måle tettleiken til systemet til å vere antal atomar delt på volumet til heile systemet. Dette vil vere ein konstant, men me vil sjå på korleis denne vil endre seg som ein funksjon av konstanten b som me nyttar ved konstruksjon av einingscellene til systemet. Storleiken på systemet vil vere gjeve ved

$$V = S_x S_y S_z,$$

kor V er volumet til systemet og S_x, S_y og S_z er lengda til kvar av sidene til systemet. Då finner me tettleiken ved

$$\rho_N = \frac{N}{V},$$

kor N er antal atomar i systemet.

Trykk

Trykket til systemet er gjeve ved formelen

$$P = \rho_N k_B T + \frac{1}{3V} \left\langle \sum_{i=1}^{N} \mathbf{F}_i \cdot \mathbf{r}_i \right\rangle,$$

kor ρ_N er tettleiken, k_B er Boltzmann konstanten, T er temperaturen, V er volumet til systemet og $\mathbf{F}_i$ og $\mathbf{r}_i$ er henholdsvis krafta og posisjonen til atom i. Då me nyttar Newtons tredje lov (N3L) i kraftutrekninga vår vil me kunne forenkle dette uttrykket og istaden rekne ho ut inne i kraftløkka. Då

$$\mathbf{F}_i \cdot \mathbf{r}_i = \sum_{j=1,\ j \neq i}^{N} \mathbf{F}_{ij} \cdot \mathbf{r}_{ij},$$

kor $\mathbf{F}_{ij}$ og $\mathbf{r}_{ij}$ er krafta og avstanden mellom atom i og j. Me set dette inn i uttrykket for trykk.

$$P = \rho_N k_B T + \frac{1}{3V} \left\langle \sum_{i=1}^{N} \sum_{j=1,\ j \neq i}^{N} \mathbf{F}_{ij} \cdot \mathbf{r}_{ij} \right\rangle.$$

No nyttar me N3L til å sjå at det held å rekne ut krafta for kvar i og j ein gong viss me nyttar at $\mathbf{F}_{ij} = -\mathbf{F}_{ji}$. Då vil me kunne stryke mange ledd mot einannan då dei vil bli rekna ut to gonger. Istaden skriv me det om til

$$P = \rho_N k_B T + \frac{1}{3V} \left\langle \sum_{i>j}^{N} \mathbf{F}_{ij} \cdot \mathbf{r}_{ij} \right\rangle.$$

Varmekapasitet

Varmekapasiteten er eit mål på kor mykje varme eit materiale kan lagre medan det skjer ei temperaturendring. Me kan rekne ut varmekapasiteten ved å nytte variansen i kinetisk energi. Me har då eit forhold som seier

$$\left\langle E_k^2 \right\rangle - \left\langle E_k \right\rangle^2 = \frac{3k_B T^2}{2N}\left(1 - \frac{3k_B}{2C_V}\right),$$

kor k_B er Boltzmann konstanten, T er temperatur og C_V er varmekapasiteten. Me vil omforme denne likninga slik at me får varmekapasiteten for seg sjølv på venstre sida. Me får då

$$\begin{aligned}
&\left\langle E_k^2 \right\rangle - \left\langle E_k \right\rangle^2 = \frac{3k_B T^2}{2N} - \frac{9k_B^2 T^2}{4NC_V}, \\
\Rightarrow \qquad &-4NC_V\left(\left\langle E_k^2 \right\rangle - \left\langle E_k \right\rangle^2 - \frac{3k_B T^2}{2N}\right) = 9k_B^2 T^2, \\
\Rightarrow \qquad &C_V = -\frac{9k_B^2 T^2}{4N\left(\left\langle E_k^2 \right\rangle - \left\langle E_k \right\rangle^2 - \frac{3k_B T^2}{2N}\right)}.
\end{aligned}$$

Me vil måle denne verdien i eininga $\frac{\mathrm{J}}{\mathrm{Kmol}}$ for å samanlikne med eksperimentelle verdiar. Boltzmann konstanten er gjeve ved

$$k_B \sim \frac{\mathrm{m^2 kg}}{\mathrm{s^2 K}} \sim \frac{\mathrm{J}}{\mathrm{K}}.$$

Då ser me at vermakapasiteten vil vere gjeve ved

$$C_V \sim \frac{\left(\frac{\mathrm{J}}{\mathrm{K}}\right)^2 \mathrm{K}^2}{\mathrm{J}^2 - \mathrm{J}^2 - \frac{\mathrm{J}}{K}\mathrm{K}^2}$$

Finn ut av denne...

Berendsen termostat

Siden temperaturen fluktuerer vil me gjerne ha moglegheit til å styre ho. Me vil implementere ein **Berendsen termostat**. Ho er gjeve ved

$$\gamma = \sqrt{1 + \frac{\Delta t}{\tau}\left(\frac{T_{\mathrm{bath}}}{T} - 1\right)}.$$

Skaleringsfaktoren γ vil me gonge med hastigheita $\mathbf{v}$ til atoma i systemet. Her er τ ein parameter som avgjer kor kjapt temperaturen skal endrast. T_{bath}

er den ønska temperaturen medan T er den noverande temperaturen. Me kan då sjå at $\gamma = 1$ for $T = T_{\text{bath}}$. Desverre er ikkje dette ein veldig realistisk termostat då temperaturen endrer seg mykje fortare enn det ville skjedd i virkelegheita. Dette gjer blant anna at me ikkje vil måle fysiske eigenskapar så lenge termostaten er på. Me vil difor nytte ho for å nå ei ynskja temperatur for så å la systemet få svinge litt før me byrjar å måle verdiar.

Algoritmar

Simulering av eit uendeleg stort system

Då me ikkje kan simulere eit uendeleg stort system vil me nytte nokre metodar kor me slepper å sjå at atoma våre forsvinn ut i ingenting.

Periodiske randbetingelsar

Viss eit atom har ein posisjon som ligg utanfor storleiken på systemet vårt vil me flytte atomet slik at det kjem inn frå den andre sida av systemet. For eit stort system kan dette vere ei god tilnærming då det heile tida forsvinn atom ut frå eit lite område, men samstundes kjem det inn nye atom slik at tettheten er jamn. Dette vil og vere med på å gje oss bevaring av total energi då det ikkje går noko tap til vegger og liknande. Periodiske randbetingelsar vil vere gjeve ved formelen

$$\mathbf{r}_i = x_i\mathbf{i} + y_i\mathbf{j} + z_i\mathbf{k}, \qquad i \in [1, N],$$

kor N er antal atom i systemet. Då vil kvar komponent i $\mathbf{r}_i$ vere gjeve ved

$$x = \begin{cases} x + S_x, & x < 0 \\ x, & x \in [0, S_x) \\ x - S_x, & x \geq S_x \end{cases},$$

kor S_x er storleiken på systemet i x-retning og x er noverande posisjon for atomet. Me gjentek dette for dei to andre retningane og.

Avstand frå endepunkta

Frå analogien ved bruk av periodiske randbetingelsar må me ta hensyn til at avstand mellom atompar ikkje nødvendigvis stemmer. Argumentet vårt er at atoma som ligg heilt i kanten av systemet har "tvillingar" som kjem inn frå andre sida på nøyaktig same tid og med dei same parametrane som seg sjølve. Når me då rekner avstand mellom atoma vil me sjekke om eigentleg ligg mykje nærare viss det eine atomet vert flytta over på den andre sida av systemet. Me kjem til å nytte "Minimum Image Criterion" for å sjekke om dette er oppfylt. Me definerer

$$\Delta\mathbf{R} = \mathbf{r}_i - \mathbf{r}_j, \qquad i \neq j, \qquad i, j \in [1, N],$$

som gjer oss avstanden mellom atoma slik det framstår i simuleringa. For å simulere eit uendeleg system vil me difor sjekke om denne oppfyller krava

$$\Delta\mathbf{R} = \Delta X\mathbf{i} + \Delta Y\mathbf{j} + \Delta Z\mathbf{k},$$

kor me har skrive $\Delta\mathbf{R}$ på komponentform. Då får me

$$\Delta X = \begin{cases} \Delta X - S_x, & \Delta X > \frac{S_x}{2} \\ \Delta X, & \Delta X \in \left(-\frac{S_x}{2}, \frac{S_x}{2}\right] \\ \Delta X + S_x, & \Delta X \leq -\frac{S_x}{2} \end{cases}$$

Velocity Verlet

Idet systemet vårt er satt opp med ein starthastighet for kvart atom vil me tidsintegrere oss fram i tid for å sjå korleis det utvikler seg. Då systemet vårt er pakka inn med periodiske randbetingelser og biletekriterie vil me nytte ein algoritme som bevarer energien. Velocity Verlet er ein **symplektisk integrator** som bevarer total energien (Hamiltonian er bevart, men denne tilsvarer ofte total energien i eit system). Velocity Verlet er i tillegg ein enkel og kompakt algoritme med eit feilledd som går som $\mathcal{O}(\Delta t^2)$. Integrasjonen består av tre ledd

$$\begin{aligned} \mathbf{v}(t + \Delta t/2) &= \mathbf{v}(t) + \frac{\mathbf{F}(t)}{m}\frac{\Delta t}{2}, \\ \mathbf{r}(t + \Delta t) &= \mathbf{r}(t) + \mathbf{v}(t + \Delta t/2)\Delta t, \\ \mathbf{v}(t + \Delta t) &= \mathbf{v}(t + \Delta t/2) + \frac{\mathbf{F}(t + \Delta t)}{m}\frac{\Delta t}{2}, \end{aligned}$$

kor me må rekne ut krafta ein gong for fyrste ledd før me byrjar å integrere oss fram i tid.

Cellelister

Det som er kostbart ved Velocity Verlet er utrekninga av kraft to gonger per ledd. Viss me rekner ut krafta mellom kvart atom vil me ha ei algoritme som går som $\mathcal{O}(N^2)$, kor N er antal atom. Ved å sette opp cellelister vil me dele opp systemet vårt i mindre celler kor kvar side av cellene har lengde r_{cut} runda av til næraste heiltal. Ved å gjere dette slipper me å leite gjennom alle atoma i systemet vårt og rekne ut krafta mellom dei. Me vil heller rekne ut krafta mellom atoma i ei celle og alle nabocellene til denne cella. Viss me då i tillegg sjekker om atoma i nabocellene ligg utanfor r_{cut} vil me redusere køyretida til $\mathcal{O}(N)$.

Algoritme 1 Rekn ut krafta ved hjelp av cellelister

```
1: for i ∈ N_x do
2:     for j ∈ N_y do
3:         for k ∈ N_z do
4:             Cell_1 = CellIndex(i, j, k)
5:             for cx ∈ [i − 1, i + 1] do
6:                 for cy ∈ [j − 1, j + 1] do
7:                     for cz ∈ [k − 1, k + 1] do
8:                         applyPeriodic(cx, cy, cz)
9:                         Cell_2 = CellIndex(cx, cy, cz)
10:                        for m ∈ Cell_1.getAtoms() do
11:                            for n ∈ Cell_2.getAtoms() do
12:                                Rekn ut krafta
13:                            end for
14:                        end for
15:                    end for
16:                end for
17:            end for
18:        end for
19:    end for
20: end for
```

Her er N_x, N_y og N_z antal celler i henholdsvis x-, y- og z-retning. cx, cy og cz viser til koordinatane til nabocellene. Metoden $applyPeriodic()$ sjekker om cellene i endepunkta har naboar på andre sida av systemet.
For kvart tidssteg må me legge atom i dei riktige cellene. Me lagrar cellene på indeksar gjeve ved formelen

$$index = iN_yN_z + jN_z + k,$$

kor i, j og k er indeksane me sender inn. Då kan me finne kva celle eit atom høyrer til i ved å nytte formelen

$$index_x = \left\lfloor \frac{x_{\text{Atom}}}{S_x} N_x \right\rfloor,$$

og tilsvarande for y- og z-retning. Då vil $index$-formelen over returnere cella som inneheld det gjeldane atomet.

Struktur

I dette prosjektet har me nytta objektorientering for å halde styr på simuleringa vår. Simuleringane vert køyrde i `C++`. Me har bygd eit rammeverk med `Python` som tek seg av køyringa i `C++` for å gje eit betre overblikk. Litt diverse `bash`-script rydder og strukturer utskriftar.

Objektar i C++

Under fylgjer ein kort oversikt over objekt me nyttar og formålet deira.

Atom

Filene `atom.h` og `atom.cpp` utgjer ein liten klasse kor me lagrar masse, posisjon, hastighet og krafta til eit atom. I tillegg gjer me kvart atom ein indeks som gjer det mogleg for oss å nytte Newtons tredje lov kor me kun vil rekne ut krafta mellom para viss atom-objektet me har valgt har ein større indeks enn det andre atom-objektet me har valgt.

Berendsen

Programma `berendsen.h` og `berendsen.cpp` implementerer Berendsen termostaten beskrevet i seksjonen om fysikken bak molekylær dynamikken. Ho vert nytta av integratoren til å skalere hastigheita viss me vil ha kontroll over temperaturen.

Cell

Programma `cell.h` og `cell.cpp` vert nytta til å ta vare på atoma som ei einskild celle inneheld. Formålet med klassa er fyrst og fremst å kunne indeksere cellene slik at me kun treng å rekne ut krafta mellom ei og ei celle ein gong. Me slepp då å sjekke for kvart atom to for-løkker djupare.

Cell List

Programma `celllist.h` og `celllist.cpp` inneheld ein vektor med Cell-objektar. Ho vil i tillegg lagre antal celler i x-, y- og z-retning. Klassa tek seg og av sorteringa av atom og legg dei inn i riktige celler.

IO

Programma `io.h` og `io.cpp` tek seg av utskrift til fil i `.xyz`-formatet. Dette formatet vert lese av VMD for visualisering av atoma.

Statistics Sampler

Programma `statisticssampler.h` og `statisticssampler.cpp` inneheld metodar som lagrar og skriv ut dei fysiske eigenskapane me vil måle til fil.

System

Programma `system.h` og `system.cpp` er sjølve fundamentet blant objekta. Klassa handterer oppsett, køyring og lagrar atom. Ho handterer sjølve gjennomføringa slik at eit main-program kun treng å køyre nokre få enkle metodar.

Unit Converter

Programma `unitconverter.h` og `unitconverter.cpp` skalerer einingane på dei fysiske verdiane slik at me arbeider me meir handterlege tal. Dette gjer og moglegheita for store numeriske avrundingsfeil mykje mindre.

Integrator

Programma `integrator.h` og `integrator.cpp` utgjer ein superklasse som tillater oss å nytte forskjellige numeriske integrasjonsalgoritmar.

Velocity Verlet

Programma `velocityverlet.h` og `velocityverlet.cpp` er ein numerisk integrasjonsalgoritme som står forklart i algoritme seksjonen.

Random

Programma `random.h` og `random.cpp` utgjer ein random-generator som me nyttar når me set starthastigheita til kvart atom med Maxwell-Boltzmann distribusjon.

Vec3

Programma `vec3.h` og `vec3.cpp` er ein lett vektorklasse som implementerer vektoroperasjoner for tre-dimensjonale vektorar. Styrken til klassa er at me berre lagrar tre double-verdiar. Ho vil difor vere kjapp under runtime.

Potential

Programma `potential.h` og `potential.cpp` er ein superklasse, som på same måte som `integrator.h` tillater bruk av forskjellige potensial. Klassa lagrar og den potensielle energien og summen av krafta og avstanden mellom atoma som me treng for å rekne ut trykket.

Lennard Jones

Programma `lennardjones.h` og `lennardjones.cpp` implementerer Lennard Jones potensialet og uttrykket for krafta. Her ligg og implementasjonen av cellelister.

Python-rammeverk

Grunna storleiken på programmet og dei forskjellige simuleringane me vil gjere er det enklare å køyre main-metoden frå `Python`. Rammeverket består av ein klasse som køyrer kommandolinje-argument. Ho tek seg av køyring av Makefile og plottar og skriv ut til fil.

Resultat

Feilestimat