Universitetet i Oslo, Fysisk Institutt
FYS3150, Prosjekt 5

Molekylærdynamikk

Live Wang Jensen

November 28, 2016

Målet med dette prosjektet er å implementere en algoritme som bruker molekylærdynamikk slik at vi kan studere ulike termodynamiske egenskaper til grunnstoffet argon. Atomene i systemet vårt er ordnet i en kubisk flatesentrert gitterstruktur, som er den strukturen argon har som fast stoff. Vi har sett på et system med 500 atomer, og simulert systemets tidsutvikling ved ulike initialtemperaturer. I tillegg har vi sett på ulike termodynamiske størrelser som potensiell og kinetisk energi, temperaturutvikling og diffusjonskonstant. Ved å plotte diffusjonskonstanten som funksjon av temperatur, fant vi ut at systemets smeltevarme lå mellom 350 og 390 Kelvin.

Innholdsfortegnelse

1 Introduksjon

Molekylærdynamikk, eller MD, er en numerisk metode som brukes for å kunne beregne ulike fysiske størrelser. Dette gjøres gjerne gjennom datasimulasjon av bevegelsen til atomer. Det finnes flere ulike MD algoritmer, den enkleste av dem bruker Newtonsk fysikk og ser på atomene som punktpartikler. MD er en svært nyttig metode som brukes innen nesten alle vitenskapelige fagfelt, fra biokjemi og materialteknologi til nanoteknologi og biofysikk. Slike simuleringer brukes blant annet til å studere oppførselen til systemer som ikke kan observeres direkte gjennom eksperimenter, som for eksempel vekst av tynnfilmer. Innenfor biofysikken bruker man ofte MD til å raffinere tredimensjonale proteinstrukturer og andre makromolekyler ut ifra eksperimentelle data hentet fra NMR spektroskopi. MD kan også brukes til å undersøke fysiske egenskaper i nanoteknologiske apparater som ennå ikke har blitt produsert.

I denne oppgaven skal vi se nærmere på grunnstoffet argon. Vi skal gradvis bygge opp en kode som til slutt gjør oss i stand til å simulere ulike egenskaper ved argon ved bruk av molekylærdynamikk. Vi starter med å se på argons gitterstruktur og hvordan denne kan implementeres i koden vår. Dette gjør slik at atomene i systemet vårt plasseres ordnet, og ikke tilfeldig inne i enhetscellen. Videre innfører vi periodiske grensebetingelser, som fører til at vi får et lukket system hvor den totale energien er konstant. Vi skal bruke Lennard-Jones potensialet til å beregne kraften som virker mellom atomene. I tillegg skal vi se nærmere på andre fysiske størrelser som kinetisk og potensiell energi, temperatur og diffusjonskonstant. Systemet vårt består av 500 atomer, og vi skal bruke programmet Ovito til å se på hvordan disse vil oppføre seg som funksjon av tid ved ulike initialtemperaturer. Dette kan brukes som et røft estimat til å finne systemets smeltepunkt. Et mer nøyaktig estimat finner vi ved å se på diffusjonskonstanten som funksjon av temperatur.

2 Teori og implementasjon

2.1 Gitterstruktur

Gitteret vårt består at *enhetsceller*, en gruppe med atomer. Større systemer vil altså bestå av flere enhetsceller etter hverandre. Argon har en kubisk flatesentrert gitterstruktur, også kalt FCC, se figur 2.1.

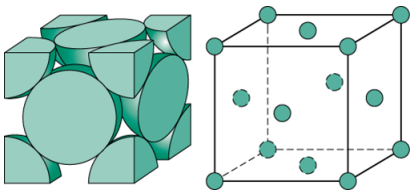

Figure 2.1: To mulige fremstillinger av gitterstrukturen FCC.

Én slik FCC celle har størrelsen b Ångstrøm, hvor b er gitterkonstanten. Vi bruker $b =$

5.26 Ångstrøm, som er den vanlige cellestørrelsen i Argon krystallen. Hver celle består av fire atomer med koordinatene

$$\mathbf{r}_1 = 0\hat{i} + 0\hat{j} + 0\hat{k} \tag{2.1}$$

$$\mathbf{r}_2 = \frac{b}{2}\hat{i} + \frac{b}{2}\hat{j} + 0\hat{k} \tag{2.2}$$

$$\mathbf{r}_3 = 0\hat{i} + \frac{b}{2}\hat{j} + \frac{b}{2}\hat{k} \tag{2.3}$$

$$\mathbf{r}_4 = \frac{b}{2}\hat{i} + 0\hat{j} + \frac{b}{2}\hat{k} \tag{2.4}$$

Vi kan nå lage $N_x \times N_y \times N_z$ slike enhetsceller ved siden av hverandre for å danne et større system, hvor origo til enhetscelle (i, j, k) er

$$\mathbf{R}_{i,j,k} = i\hat{\mathbf{u}}_1 + j\hat{\mathbf{u}}_2 + k\hat{\mathbf{u}}_3, \tag{2.5}$$

hvor $i = 0, 1, ..., N_x - 1, j = 0, 1, ..., N_y - 1$, og $k = 0, 1, ..., N_z - 1$. Enhetsvektorene til enhetscellene skaleres med gitterkonstanten b slik at

$$\hat{\mathbf{u}}_1 = b\hat{\mathbf{i}}, \quad \hat{\mathbf{u}}_2 = b\hat{\mathbf{j}}, \quad \hat{\mathbf{u}}_3 = b\hat{\mathbf{k}}. \tag{2.6}$$

Siden gitterkonstanten $b = 5.26$ Ångstrøm, og vi har fire atomer per celle, ender vi opp med en tetthet på $\rho = m/V = 4m_{Ar}/b^3 = 1822, 8kg/m^3$.

2.2 Periodiske grensebetingelser

Vi bruker såkalte **periodiske grensebetingelser** når vi simulerer systemet vårt. Når vi simulerer fysiske systemer på datamaskinen, vil disse være begrenset av maskinens kapasitet. Typiske MD simuleringer inneholder ofte flere millioner atomer på et område mindre enn én kubikkmikrometer. Vi kan komme unna det som skjer i enden av denne kuben ved å bruke periodiske grensebetingelser. På denne måten kan vi simulere store (uendelige) systemer ved en enkelt celle. Når et atom beveger seg forbi veggen på den ene siden, vil den dukke opp igjen på motsatt vegg med samme hastighet, se figur 2.2. Store systemer består at et uendelig antall slike enhetsceller, hvor én av disse er vår originale simulering, mens resten er kopier, eller *bilder*, av denne. I tre dimensjoner vil vi altså ha én original celle, med 26 naboer, som er kopier av denne. Vi trenger derfor kun å simulere hva som skjer i den originale enhetscellen, og dette sparer oss enormt med beregningstid -og krefter. Dette kan gjøres ved den enkle algoritmen

```
if (periodic_x) then
  if (x <  -x_size * 0.5) x = x + x_size
  if (x >=  x_size * 0.5) x = x - x_size
end if
```

hvor `x_size` er lengden på boksen i én retning, og `x` er posisjonen til atomet i denne retningen. I tillegg må avstanden mellom atomene følge *minimum image* kriteriet

```
if (periodic_x) then
  dx = x(j) - x(i)
  if (dx <= -x_size * 0.5) dx = dx + x_size
  if (dx >  x_size * 0.5) dx = dx - x_size
endif
```

hvor `dx` er avstanden mellom atom i og j. Vi vet at i tre dimensjoner så vil hvert atom eksistere 27 ganger, en gang i originalcellen og 26 ganger i nabocellene. Når vi beregner kraften som virker mellom atom i og j, velger vi altså den kopien av atom j som ligger *nærmest* atom i. Siden vi jobber i tre dimensjoner, må vi gjøre samme prosedyre i y -og z-retning. De eneste kreftene som virker i systemet vårt er indre krefter, slik at den totale energien er konstant. De periodiske grensebetingelsene fører til at systemet ikke kan utveksle atomer med miljøet rundt, som betyr at vi kan kalle dette for et *mikrokanonisk ensemble*.

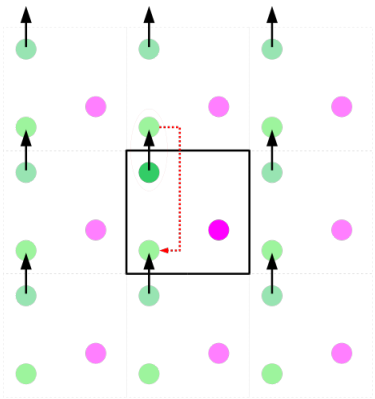

Figure 2.2: Periodiske grensebetingelser i 2D. Her har vi én original enhetscelle, og åtte kopier av denne.

2.3 Lennard-Jones potensialet

Vi kommer til å bruke Lennard-Jones potensialet til å beregne energien mellom to atomer i og j

$$U(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right], \tag{2.7}$$

hvor $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ er avstanden mellom atom i og atom j, ϵ er dybden på potensialbrønnen (med enhet energi) og σ er avstanden når potensialet er null.

Lennard-Jones potensialet er vanlig å bruke når vi skal beskrive kreftene som virker mellom to nøytrale atomer eller molekyler. Argon er en edelgass og vil derfor ikke danne kovalente bindinger mellom atomene. I ligning (2.7) ser vi et ledd r^{-12}. Dette leddet

beskriver de frastøtende kreftene, som kommer av at atomene frastøter hverandre når avstanden mellom dem er liten, siden deres elektronbaner vil overlappe. Dette kalles den steriske effekten, som fører til at energien øker brått når avstanden blir mindre. Dette illustreres i figur 2.3. Leddet r^{-6} beskriver den tiltrekkende kraften, som virker ved lengre avstander. Her virker van der Waals krefter, som er en relativt svak kraft, i forholdt til kovalente bindinger.

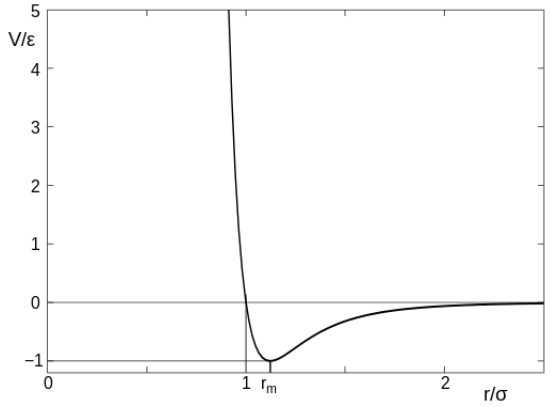

Figure 2.3: Graf som viser styrken på potensialet som funksjon av avstand mellom atomene. r_m er avstanden når potensialet har verdien $-\epsilon$.

Siden vi jobber med argon, bruker vi

$$\frac{\epsilon}{k_B} = 119.8\text{K}, \quad \sigma = 3.405\text{Angstrom}. \tag{2.8}$$

I dette prosjektet brukes såkalte MD enheter. Definisjonen av disse finnes i Appendix A. Potensialet gitt ovenfor gir oss termodynamiske størrelser som stemmer godt med argons eksperimentelle dataer ved ekvilibrium. Systemets tilstandsligning er gitt ved van der Waals ligning. Systemets totale potensielle energi V finner vi ved å summe over alle atompar, hvor vi teller hvert par kun én gang

$$V = \sum_{i>j} U(r_{ij}) \tag{2.9}$$

Kraften finner vi ved

$$\mathbf{F}(r_{ij}) = -\nabla U(r_{ij}). \tag{2.10}$$

x-komponenten av kraften blir da

$$F_x(r_{ij}) = -\frac{\partial U}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial x_{ij}}. \tag{2.11}$$

y -og z-komponenten finner vi på samme måte. Ved å bruke at $r = \sqrt{x^2 + y^2 + z^2}$, kan vi derivere dette uttrykket analytisk, slik at

$$\frac{\partial U}{\partial r_{ij}} = 4\epsilon \left[-12 \left(\frac{\sigma^{12}}{r_{ij}^{13}} \right) + 6 \left(\frac{\sigma^{6}}{r_{ij}^{7}} \right) \right] \tag{2.12}$$

og

$$\frac{\partial r_{ij}}{\partial x_{ij}} = \frac{x_{ij}}{r_{ij}}, \tag{2.13}$$

slik at

$$F_x(r_{ij}) = -4\epsilon \left[-12 \left(\frac{\sigma^{12}}{r_{ij}^{13}} \right) + 6 \left(\frac{\sigma^{6}}{r_{ij}^{7}} \right) \right] \cdot \frac{x_{ij}}{r_{ij}}. \tag{2.14}$$

Tilsvarende kan gjøres for $F_y(r_{ij})$ og $F_z(r_{ij})$. Når vi implementerer dette i koden vår, bruker vi **Velocity Verlet** metoden som integrator. Vi finner altså atomenes hastighet og posisjon ved å bruke at

$$\vec{v}(t + dt/2) = \vec{v}(t) + \frac{\vec{F}(t)}{m}\frac{dt}{2}$$
$$\vec{r}(t + dt) = \vec{r}(t) + \vec{v}(t + dt/2)dt$$
$$\vec{v}(t + dt) = \vec{v}(t + dt/2) + \frac{\vec{F}(t + dt)}{m}\frac{dt}{2}$$

hvor dt er tidssteget vårt.

2.4 Kinetisk energi, temperatur og diffusjonskonstanten

Vi har også beregnet flere ulike fysiske størrelser som kinetisk energi, total energi, temperatur og diffusjonskonstant for å kunne se hvordan systemet utvikler seg ved ulike initialtemperaturer.
Systemets kinetiske energi er

$$E_k = \sum_{i=1}^{N_{atomer}} \frac{1}{2} m_i v_i^2 \tag{2.15}$$

hvor m_i og v_i er massen og hastigheten til atom nummer i. Den potensielle energien er gitt i ligning (2.9), og vi finner den totale energien ved å addere den kinetiske og den potensielle energien.

$$E_{tot} = E_k + V \tag{2.16}$$

Vi vet at denne summen må være konstant, siden vi jobber med et mikrokanonisk system. I tillegg skal den totale bevegelsesmengden i systemet være null. Dette er derfor en god måte å teste om programmet virker som det skal.
Vi kan kalkulere et estimat for temperaturen ved hjelp av ekvipartisjonsteoremet

$$\langle E_k \rangle = \frac{3}{2} N_{atomer} k_B T \tag{2.17}$$

som kan brukes til å definere den *momentane* temperaturen

$$T = \frac{2}{3}\frac{E_k}{N_{atomer}k_B} \tag{2.18}$$

Til slutt kan vi finne diffusjonskonstanten ved å bruke Einstein relasjonen som sier at

$$\langle r^2(t)\rangle = 6Dt \tag{2.19}$$

hvor D er diffusjonskonstanten, t er tiden og

$$r_i^2(t) = |\mathbf{r}_i(t) - \mathbf{r}_i(0)|^2 \tag{2.20}$$

hvor $\mathbf{r}_i(t)$ er posisjonen til atom i ved tiden t. Ved å øke temperaturen, vil systemet bli mer og mer kaotisk. Ved en viss temperatur vil vi miste krystallstrukturen, vi kaller dette for systemets smeltepunkt. Diffusjonskonstanten D gir oss en estimat på temperaturen ved smeltepunktet. Vi vet at D vil være omtrent null når systemet er i fast form, mens den vil øke kraftig ved smeltepunktet.

2.5 Kodens struktur

Koden som er brukt i dette prosjektet er objekt orientert, og bygger på koden skrevet av Anders Hafreager [6]. Figur 2.4 viser kodens flytskjema. Koden består av en rekke nøstede funksjoner som avhenger av hverandre. Aller først starter vi med å initialisere systemet vårt, ved å bestemme antall celler i gitteret, den initielle temperaturen og gitterkonstanten, som bestemmer systemets tetthet. Deretter konverteres enhetene i klassen `unitconverter.cpp`, slik at ulike fysiske størrelser som masse, lengde og energi har en enhet som er tilpasset systemet vi jobber med. Deretter setter vi opp selve gitteret ved å plassere partiklene i en tredimensjonal kubisk flatesentrert gitterstruktur, akkurat som i argons krystallstruktur. Det betyr at vi har ett atom i hvert hjørne, og ett atom på hver flate. Hvert atom får en tilfeldig initialhastiget bestemt av Boltzmanns fordelingslov som funksjon av initialtemperaturen, slik at

$$P(v_i)dv_i = \left(\frac{m}{2\pi k_B T}\right)^{1/2} exp\left(-\frac{mv_i^2}{2k_B T}\right) dv_i \tag{2.21}$$

hvor m er atomets masse, k_B er Boltzmanns konstant og T er temperaturen. Dette er en normalfordeling med null i gjennomsnitt og standardavvik $\sigma = \sqrt{k_B T/m}$. I tillegg fjerner vi den totale initielle bevegelsesmengden ved å trekke fra den gjennomsnittlige initielle bevegelsesmengden per atom fra hvert av atomene. Dette fører til at systemets totale bevegelsesmengde blir null.
Videre bestemmes hvilken integrator som skal brukes; i vårt tilfelle er det `velocityverlet.cpp`. Vi er nå klare til å starte selve simuleringen.

Vi starter med funksjonen `step` som finnes i klassen `system.cpp`. Denne funksjonen starer ned å kalle på den integratoren vi har valgt. Inne i integratoren kalles det på funskjonen `calculateForces()` som finnes i `system.cpp`. Vi bruker `LennardJones` potensialet, som regner ut den potensielle kraften som funksjon av avstanden melom atom i og j, r_{ij}. Integratoren beregner atomenes nye hastighet og posisjon, basert på forrige tidssteg. For å kunne beregne kreftene er det brukt periodiske grensebetingelser. Etter dette går vi til neste tidssteg, og vi gjentar `step` funksjonen ved å loope gjennom tidsstegene. Etter hver loop registreres ulike termodynamiske størrelser som temperatur, kinetisk og potensiell energi. Posisjonen til partiklene lagres i en fil og visualiseres i Ovito.

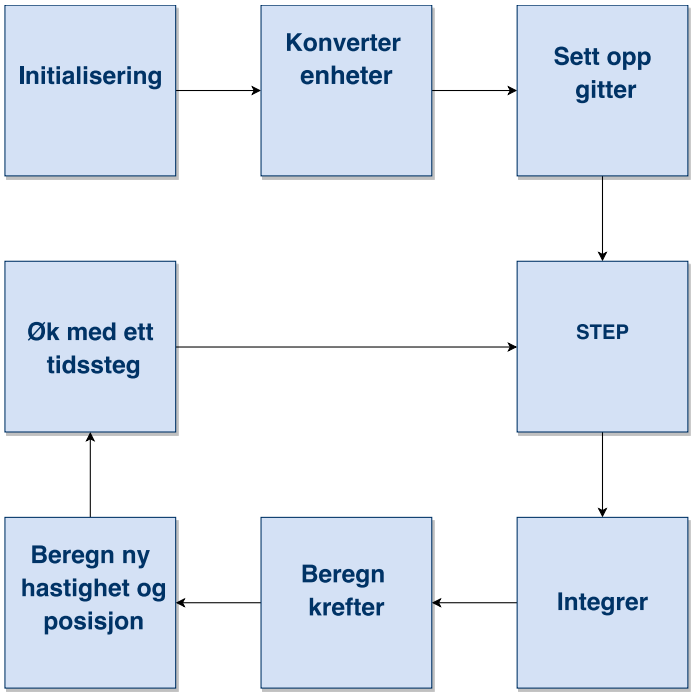

Figure 2.4: Flytskjema som beskriver kodens struktur.

3 Resultater

Figur 3.1 og 3.2 viser oss fordelignen av atomer før og etter vi implementerte gitterstrukturen FCC. Figurene er laget i Ovito, og viser fordelingen ved starten av simuleringen, før kreftene har begynt å virke.

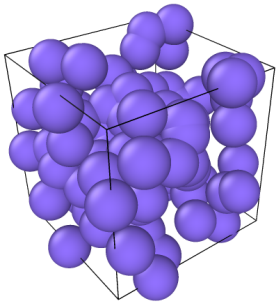

Figure 3.1: Tilfeldig plassering av atomer.

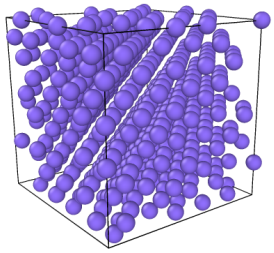

Figure 3.2: Atomer plassert i FCC gitterstruktur.

Figur 3.3, 3.4, 3.5 og 3.6 viser hvordan systemet utviklet seg ved ulike initialtemperaturer på hvv 50, 250, 450 og 700 Kelvin. Her er det brukt 5×5 × 5 enhetsceller, som tilsvarer 500 atomer.

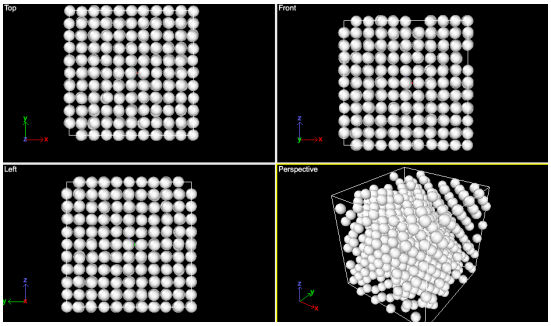

Figure 3.3: Initialtemperatur 50 K

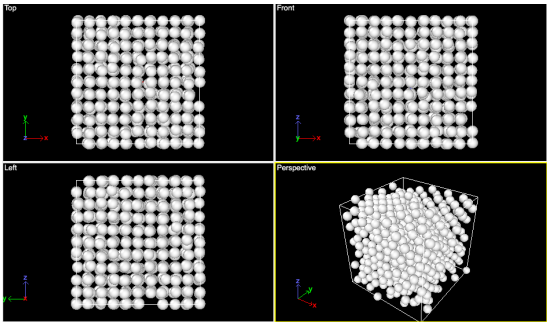

Figure 3.4: Initialtemperatur 250 K

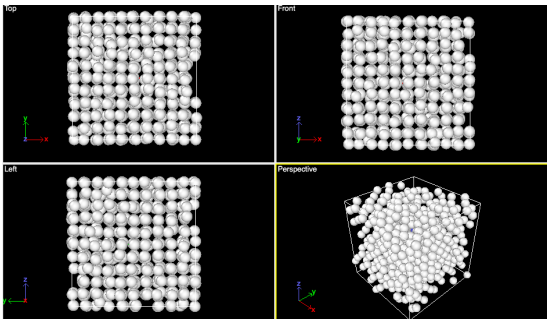

Figure 3.5: Initialtemperatur 450 K

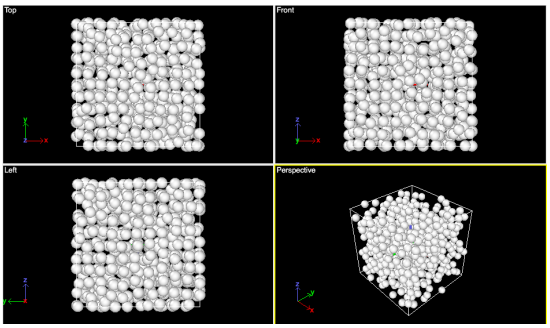

Figure 3.6: Initialtemperatur 700 K

I figur 3.7 og 3.8 ser vi hvordan temperaturen og energien utvikler seg som funksjon av antall tidssteg.

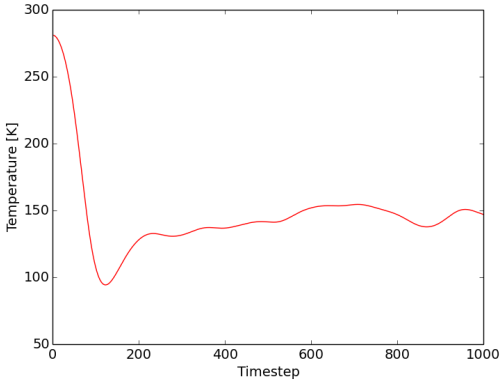

Figure 3.7: Temperatur som funksjon av antall tidssteg. Her er ett tidssteg det samme som 10^{-15} sekunder.

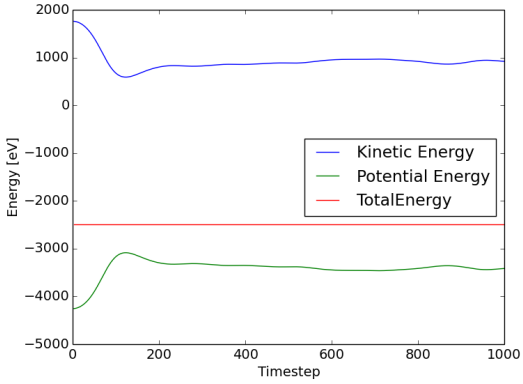

Figure 3.8: Potensiell, kinetisk og total energi som funksjon av antall tidssteg.

I figuren under ser vi T/T_{init} som funksjon av T_{init}, hvor T er temperaturen til systemet etter det har nådd termisk likevekt, og T_{init} er initialtemperaturen som systemet starter med.

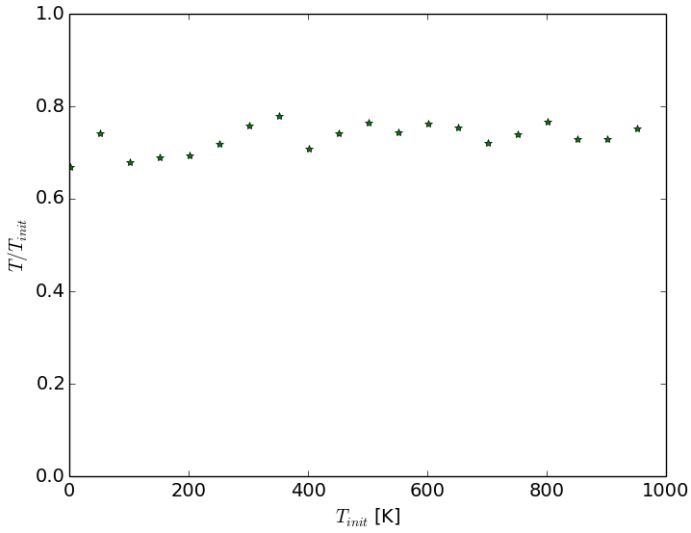

Figure 3.9: Forholdet mellom temperatur og initialtemperatur som funksjon av initialtemperatur holder seg tilnærmet konstant.

Figur 3.10 viser diffusjonskonstanten som funksjon av systemets endelige temperatur T når vi starter simuleringen med ulike initialtemperaturer T_{init}.

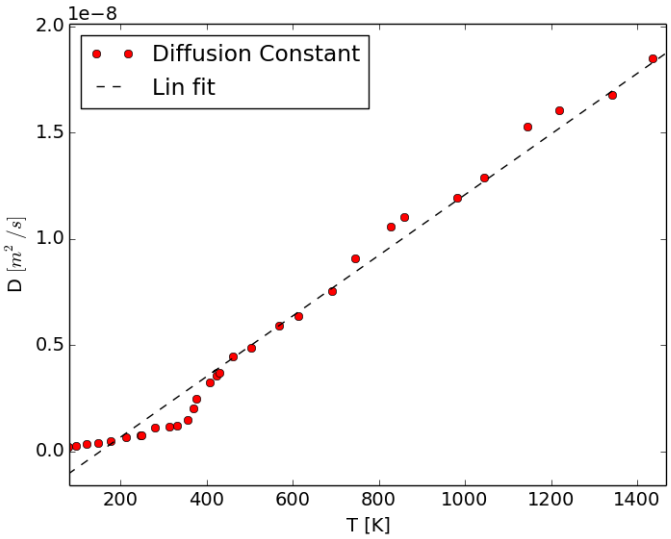

Figure 3.10: Diffusjonskonstanten ved ulike temperaturer. Den starter med å være nær null før den plutselig får en brå økning. Dette kan brukes til å finne systemets smeltetemperatur.

4 Diskusjon

I figurene 3.3, 3.4, 3.5 og 3.6 ser vi hvordan systemet utvikler seg ved ulike initialtemperaturer. Som forventet, ser vi at systemet blir mer kaotisk ved høyere initialtemperaturer. Ved såpass høye temperaturer (eks 700 Kelvin) vil ikke atomene være bundet til sin faste plass i krystallstrukturen, og systemet vårt blir flytende. Vi ser at ved en initialtemperatur på 50 K har vi fortsatt beholdt krystallstrukturen. Allerede ved en initialtemperatur på 250 K begynner det å bli vanskelig å se krystallstrukturen tydelig. Det ville være vanskelig å bestemme ved hvilken temperatur systemet vårt går over til å bli væske bare ved å se på disse bildene. Vi bruker derfor diffusjonskonstanten D til å få et bedre estimat av smeltepunktet til systemet.

Temperaturutviklingen til systemet vises i figur 3.7. Vi ser at systemet starter med en initialtemperatur på 300 K. Graften starter med å synke brått i løpet av de første hundre tidsstegene. Vi vet at temperaturen er linket til atomenes kinetiske energi. Jo høyere kinetisk energi atomene har, jo høyere blir temperaturen. Vi ser i figur 3.8 at også den kinetiske energien dropper i starten av simuleringen. Etter at temperaturen har droppet, stabiliserer temperaturen seg på omtrent halvparten av verdien til initialtemperaturen. Det at temperaturen har stabilisert seg tyder på at systemet har nådd likevekt.

I figur 3.8 ser vi kinetisk, potensiell og total energi i samme plott. Når vi så på verdien til totalenergien, oppdaget vi en liten fluktuasjon i det andre desimaltegnet. Dette kommer

av avrundingsfeil og feil i Velocity Verlet integratoren. Vi kan derfor se bort fra disse fluktuasjonene. Den totale energien holder seg altså konstant, som forventet. Dette viser at koden vi har skrevet fungerer i henhold til de fysiske lovene. Den kinetiske energien faller også brått i starten, i likhet med temperaturen. Samtidig som den kinetiske energien faller, øker den potensielle energien. Det kommer av energibevaringen, da den totale energien må forbli konstant. Den hurtige økningen i potensiell energi kommer av at systemet mister sin krystallstruktur, hvor den potensielle energien er lavest.

Ved å simulere tidsutviklingen til systemet ved ulike initialtemperaturer T_{init}, kunne vi finne temperaturen T ved termisk likevekt. I 3.9 er forholdet T/T_{init} plottet mot T_{init}. Dette folholdet holder seg nesten konstant ved 0.7. Dette viser at temperaturen ved likevekt er omtrent 70% av initialtemperaturen. For at systemet skal oppnå en endelig temperatur ved likevekt, spiller det altså ingen rolle hvilken initialtemperatur vi bruker, siden dette forholdet er konstant.

I figur 3.10 ser vi diffusjonskonstanten som funksjon av temperatur. Vi vet at denne skal være omkring null når systemet befinner seg i fast stoff tilstand. Det kommer av at atomenes forflytning $\langle r^2(t)\rangle$ fra deres initialposisjon er ganske liten, og siden $D = \langle r^2(t)\rangle/6t$, får vi at D blir nær null. Når temperaturen er mellom 350 K og 390 K har vi en kraftig økning i D. Det betyr at systemets smeltepunkt må ligge et sted i intervallet $350K < T < 390K$. Systemet går da over til væskeform, og D fortsetter å øke lineært med temperaturen.

5 Konklusjon

I løpet av dette prosjektet har vi klart å implementere et program som bruker molekylærdynamikk til å simulere grunnstoffet Argon. Vi har sett på hvordan dette systemet oppfører seg ved ulike initialtemperaturer. Ved å beregne ulike størrelser som potensiell energi, kinetisk energi, temperatur og diffusjonskonstant, kan vi se på hvordan disse vil utvikle seg som funksjon av tid. Vi har sett på gittere med størrelsen 5×5×5, og fått bekreftet at koden fungerer ved hjelp av enkle tester som at den totale energien skal være bevart, og dermed konstant.
Vi har brukt programmet Ovito til å observere krystallstrukturen og hvordan atomenes posisjon endret seg som funksjon av temperatur og tid. Det viste seg å være vanskelig å bestemme systemets smeltepunkt bare ved å se på se på når krystallstrukturen opphørte. Vi brukte derfor diffusjonskonstanten til å estimere systemets smeltetemperatur, og at denne måtte ligge mellom 350 K og 390 K.

6 Appendix A

I programmet vårt brukes det MD enheter, hvor de fire følgende enhetene er definert ved

$$1 \text{ masseenhet } = 1 \text{ atommasse } = 1.661 \times 10^{-27}\text{kg}, \tag{6.1}$$

$$1 \text{ lengdeenhet } = 1.0\text{angstrom} = 1.0 \times 10^{-10}\text{m}, \tag{6.2}$$

$$1 \text{ energienhet } = 1.651 \times 10^{-21}\text{J}, \tag{6.3}$$

$$1 \text{ temperaturenhet} = 119.735\text{K}. \tag{6.4}$$

og vi får at Boltzmanns konstant $k_B = 1$. Ut ifra dette kan vi utlede størrelsen på andre enheter, som for eksempel tid. Ved å bruke $E = mc^2$ får vi at

$$\text{Energi} = \text{masse} \times \frac{\text{lengde}^2}{\text{tid}^2} \tag{6.5}$$

som gir oss

$$\text{Tid} = \text{lengde} \times \sqrt{\frac{\text{masse}}{\text{energi}}} \tag{6.6}$$

Setter vi inn de oppgitte verdiene for lengde, masse og energi, får vi at

$$1 \text{ tidsenhet } = 1.0 \times 10^{-10}\sqrt{\frac{1.661 \times 10^{-27}}{1.651 \times 10^{-21}}}\ \text{s} = 1.00224 \times 10^{-13}\text{s}. \tag{6.7}$$

References

[1] M. Hjort-Jensen: Computational physics, lecture notes 2015. Fysisk institutt, UiO, 2016.

[2] Oppgavetekst: Project 5, Fysisk institutt, UiO, 19.11.16

[3] Gitterstrukturer: <http://frey.no/wp-content/uploads/2014/01/3-Gitterstrukturer.pdf>, 19.11.16

[4] Molekylærdynamikk: <https://no.wikipedia.org/wiki/Molekylrdynamikk>, 19.11.16

[5] XYZ file formal: <https://en.wikipedia.org/wiki/XYZ_file_format>, 19.11.16

[6] Opprinnelig kode, A. Hafreager: <https://github.com/andeplane/molecular-dynamics-fys3150>, 19.11.16

[7] Deler av koden er hentet fra: <https://github.com/GioPede/FYS3150/tree/master/Project5>, 20.11.16

[8] Periodic boundary conditions: <https://en.wikipedia.org/wiki/Periodic_boundary_conditions>, 20.11.16

[9] van der Waals force: <https://en.wikipedia.org/wiki/Van_der_Waals_force>, 28.11.16

[10] Linear regression with matplotlib: <http://stackoverflow.com/questions/6148207/linear-regression-with-matplotlib-numpy>, 28.11.16