Undervisningsmateriale for gymnasieelever

Morten Westfall Norup Jørgensen
Carsten Svaneborg

7. november 2023

University of Southern Denmark, Department of Physics, Chemistry, and Pharmacy

Vi introducerer her en række simulationsprojekter inden for biofysik og soft-matter. PhyLife på Syddansk Universitetet er et interdisciplinært forskningsmiljø bestående af forskere både fra Fysik og Molekylær Biologi og disse projekter ligger inden for vores forskningsområde.

Biofysik er det forskningsfelt hvor vi anvender fysikkens værktøjskasse til at forstå biologiske systemer. Der er mange systemer at vælge i mellem, men vi har valgt at fokuserer på at forske i hvordan cellernes membraner virker og hvordan flydende krystallinsk orden opstår i tynde film af celler.

Soft-matter er et forskningsfelt hvor vi fokuserer på hvordan komplicerede materialer mellem fast og flydende virker. Disse er materialer der både er viskøse som væsker men også elastiske som faste stoffer. Vi kender dem fra hverdagen som geler, gummi, maling, lim, smørremidler, og flydende krystaller men de fleste materialer vi møder i køkkenet er også soft-matter f.eks. dej, kød, budding og vingummi. Naturen har også valgt at bygge vores krop med soft-matter.

Computersimulationer er en relativ ny forskningsmetode, og udgør et vigtigt kompliment til teori og eksperiment. Ved at bygge modeller af komplekse systemer og simulerere disse på computeren kan vi få viden, som kan svær at opnå med målinger. De bidrager derfor til at fortolke eksperimentelle resultater. Alle teorier laver også simplificerende antagelser, med computeren kan vi undersøge mere realistiske modeller, og derfor lære hvordan teorierne kan forbedres.

Det er vores mål med de nedenstående simulationsprojekter at give en introduktion til fænomener og fysik fra soft-matter og biofysikkens verden.

For mere information:
PhysLife: `https://phylife.sdu.dk/`

FKF: Institut for Fysik, Kemi, og Farmaci:
`https://www.sdu.dk/da/om_sdu/institutter_centre/fysik_kemi_og_farmaci`

BMB: Institut for biokemi og molekylærbiologi:
`https://www.sdu.dk/da/om_sdu/institutter_centre/bmb_biokemi_og_molekylaer_biologi`

Kapitel 1

Computerfysik

Computerfysik (på engelsk: computational physics) er en gren af fysikken hvor vi anvender computer til at lave fysik. Vi kan bygge modeller af alt fra elementar partikler til universet, og udføre simulationer af alt fra quark gluon plasmaer til hele universets udvikling siden Big Bang. Overordnet skal vi igennem følgende processer for at kunne undersøge fysikken af et system med en computer:

1. Byg en model af systemet.
2. Hvad for data ønsker vi at få ud af modellen?
3. Implementér modellen som et softwareprogram.
4. Kør programmet for at simulere forskellige systemer, mens programmet køres gemmes de data vi er interesseret at undersøge..
5. Visualiser simulationerne. Plot simulationsdata. Ved at fortolke visualiseringer og grafer lærer vi om fysikken for det system vi undersøger.

For at bygge en model skal vi vide hvad de relevante komponenter er i systemet, samt hvad er de relevante egenskab af komponenterne. Hvordan vekselvirker de, og hvad er bevægelsesligningen. F.eks. for en model af solsystemet er de relevante komponenter solen og planeterne. Den vigtigste egenskab er hvor langt de er fra solen samt hvor tunge de er. De vekselvirker vha. tyngdekræften, og bevægelsesligningen er Newtons anden lov.

Her har vi allerede bygget modellerne, og softwareprogrammet LAMMPS, som I skal anvende er installeret på SDUs supercomputer. LAMMPS skaber visualiseringer dvs. billeder og film af simulationen. Dvs. jeres fokus bliver på at udføre simulationer af forskellige systemer, plotte simulationsdata, samt at fortolke hvad dette betyder.

1.1 Molekyledynamik

Vi tager her udgangspunkt i den metode der hedder Molekyledynamik[1] (MD, på engelsk Molecular Dynamics). Vi beskriver et system, som en mængde af punkt partikler, der vekselvirker med hinanden og bevæger sig som beskrevet af Newtons anden lov. Vi kan f.eks. tænke på punkterne som atomkernerne i et molekyle, men det kan også være planeterne i solsystemet.

I MD simulationer antager vi at tid er en diskret størrelse, $t(n) = n\Delta t$, hvor n et et helt tal der fortæller hvor mange tidsskridt vi har simuleret, og Δt er længden af et enkelt tidsskridt.

Lad os regne med en 1D partikel. Dens tilstand er beskrevet ved positionen $x(n)$ og hastigheden $v(n)$ til tiden $t(n) = n\Delta t$. Opgaven er at finde partiklens tilstand i det næste tidsskridt dvs. $x(n+1)$ og $v(n+1)$ til tiden $t(n+1) = (n+1)\Delta t$.

Fra klassisk mekanik ved vi at

$$\frac{dx(t)}{dt} = v(t) \tag{1.1}$$

$$\frac{dv(t)}{dt} = a(t) = \frac{F(t)}{m} \tag{1.2}$$

Her er d/dt differentitation mht. tiden. De to ligninger er definitionen af hastighed og acceleration henholdsvis, og i den anden ligning har jeg brugt Newtons anden lov til at relatere accelerationen til kraften, som partiklen bliver påvirker af. Så længe vores tidsskridt Δt er lille nok så kan vi hastigheden og accelerationen som tilnærme:

$$\frac{dx(t)}{dt} \approx \frac{x(n+1) - x(n)}{\Delta t} = v(n) \tag{1.3}$$

$$\frac{dv(t)}{dt} \approx \frac{v(n+1) - v(n)}{\Delta t} = \frac{F(n)}{m} \tag{1.4}$$

Højresiden indeholder svaret som vi leder efter, og efter lidt algebra får vi løsningen

$$x(n+1) = v(n)\Delta t + x(n) \tag{1.5}$$

$$v(n+1) = \frac{F(n)}{m}\Delta t + v(n) \tag{1.6}$$

[1] https://en.wikipedia.org/wiki/Molecular_dynamics

Dette viser hvordan vi kan udregne den næste tilstand af en partikel i 1D. Hvis vi laver en simulation af et 3D system, så har vi istedet 6 ligninger som ovenstående, en for hver af de 3 komponenter af position og hastighed.

Tricket med disse to ligninger er at det er let for en computer at køre dem igen og igen. Kører vi ligningerne f.eks. 10^9 gange har vi forudset systemets tilstand $10^9 \Delta t$ ude i fremtiden. Det eneste det kræver er at vi kender systemets tilstand til tiden $t(0)$, dvs. $x(0)$ og $v(0)$, samt at vi hele/ tiden kan udregne hvad kraften er.

Simulationer baserer på ovenstående ligninger hedder Eulers metode[2] (efter Leonhard Euler 1707-1783). I praksis anvender vi Verlets metode[3] i vores simulationer (efter Loup Verlet 1931-2019). Ideen er den samme, men Verlets metode er mere præcis.

1.2 Modeller

For at modellere et system skal vi finde ud af hvad partiklerne repræsenterer samt hvad for kræfter, der virker mellem dem. Hvis vores punkter ovenover repræsenterer planeter i solsystemet så er det tyngdekræften mellem dem, som vi skal udregne. Men hvis punkterne repræsenterer atomer i et molekyle så er det kraften for kemiske bindinger vi skal udregne.

Som fysikere kan vi også gruppere atomerne sammen og sige at en partikel repræsenterer en kemisk gruppe, og flere grupper til sammen udgør et molekyle. Dette er en grov model for et molekyle, men den gør det lettere at forstå hvad der sker, når mange molekylet vekselvirker f.eks. i en væske.

Hvis vi f.eks. ville lære omkring fysikken bag fugleflokke, så kunne vi tænke på de enkelte fugle som partikler i en simulation. Hvordan vekselvirker fugle? De er tiltrukket af hinanden så de klumper sammen, men frastøder hinanden på kort afstand så de ikke støder sammen. Men en flok må også have en vekselvirkning så hver fugl gerne flyve i samme retning som dens naboer. Sådanne modeller hedder Vicsek modeller[4].

Nedenfor viser vi hvordan man kan bygge modeller ved at kombinere forskellige vekselvirkningspotentialer.

[2] https://en.wikipedia.org/wiki/Euler_method
[3] https://en.wikipedia.org/wiki/Verlet_integration
[4] https://en.wikipedia.org/wiki/Vicsek_model

1.3 Molekyler

Den simpleste model for en kemisk binding er en 3D fjeder, vi kan starte med at skrive potentialet

$$U(r) = \frac{1}{2}k(r-l)^2, \tag{1.7}$$

det afhænger kun af længden af fjederen r, k er fjeder konstanten, mens l er ligevægtslængden.

I en simulation har vi typisk mange partikler, som vi nummererer fra 1 til N. Men hvis to partikler nummer i og nummer j sidder sammen med en fjeder mellem sig, så er deres koordinater $\mathbf{R}_i = (x_i, y_i, z_i)$ og $\mathbf{R}_j = (x_j, y_j, z_j)$, og afstanden mellem dem er givet ved

$$r = |\mathbf{R}_i - \mathbf{R}_j| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}, \tag{1.8}$$

For at udregne x komponenten af kræften på den i'te partikel fra den j'te partikel, så skal vi differentierer potentialet med hensyn til partiklens x_i koordinat

$$\begin{aligned}
F_{xi} &= -\frac{dU(r(x_i, x_j, y_i, y_j, z_i, z_j))}{dx_i} \qquad (1.9)\\
&= -\frac{dU(r)}{dr} \times \frac{dr(x_i, x_j, y_i, y_j, z_i, z_j)}{dx_i}\\
&= -\left(\frac{d}{dr}\frac{1}{2}k(r-l)^2\right) \times \left(\frac{d}{dx_i}\left[(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2\right]^{1/2}\right)\\
&= -\left(\frac{1}{2}2k(r-l)\frac{d}{dr}(r-l)\right) \times \left(\frac{1}{2}\left[(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2\right]^{-1/2}\frac{d}{dx_i}(x_i - x_j)^2\right)\\
&= -\left(k(r-l)\right) \times \left(\frac{1}{2}\left[(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2\right]^{-1/2} 2(x_i - x_j)\frac{d}{dx_i}(x_i - x_j)\right)\\
&= -\left(k(r-l)\right) \times \left(\frac{x_i - x_j}{r}\right)\\
&= k(r-l)\left(\frac{x_j - x_i}{r}\right)
\end{aligned} \tag{1.10}$$

Den første ligning er relationen mellem konservative kræfter og deres potentialer. I den anden ligning brugte jeg kæde reglen til at simplificere udtrykket, fordi potentialet kun afhænger af r, mens r afhænger af x_i. I den tredie ligning skal jeg nu differentiere potentialet mht. afstand i den første parentes, og afstanden mht. x_i i

den anden parentes. De er begge sammensatte funktioner, så det tager et par skridt at gøre. Udtrykket i firekant parentesen er r^2, så $[r^2]^{-1/2} = r^{-1}$ derfor får vi en r i nævneren. Til sidst fører jeg fortegnet ind i den anden parentes og bytter om på i og j. For at udregne y og z komponenterne af kraften så skulle vi gennem den samme udregning, men resultatet ville blive det samme, blot med x udskiftet med y eller z.

Det var matematikken, men hvad betyder det fysisk? Husk at $r = |\mathbf{R}_i - \mathbf{R}_j|$. Dvs. at vi kan skrive den vektorielle kraft på partikel i fra partikel j som

$$\mathbf{F}_i = k(r-l)\left(\frac{\mathbf{R}_j - \mathbf{R}_i}{|\mathbf{R}_j - \mathbf{R}_i|}\right) \tag{1.11}$$

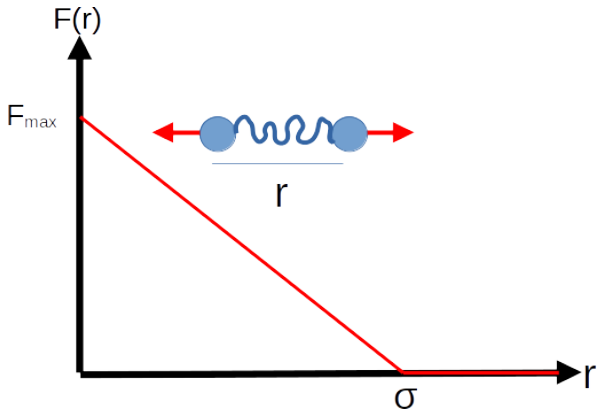
Parentesten er en enhedsvektor, der peger fra partikel i i retning af partikel j. Længden af kraften er givet ved tallet foran parentesen $|\mathbf{F}_i| = k(r-l)$. Hvis afstanden er lig med ligevægtslængden så er kraften 0. Hvis fjederen er længere end ligevægtslængden ($r > l$) så er der en positiv kraft på i i retning af j, dvs. i bliver trukket i js retning for at forkorte længden af fjederen. Hvis fjederen derimod er kortere end ligevægtslængden ($r < l$) er kraft på partikel i, der peger væk fra partikel j, dvs. en kraft der vil forlænge fjederen. Fjederen vil altså hele tiden vil prøve at etablerer ligevægtslængden mellem partikel i og j. I særtilfældet af en Hooksk fjeder er $l = 0$ og vi får $|\mathbf{F}_i| = kr$ som er Hooks lov.

Hvad med kraften på partiklen j fra partiklen i ? Vi kunne udregne F_{xj} ved at differentiere $U(r)$ mht. x_j. Den eneste forskel ville være at $d(x_i - x_j)/dx_i = +1$ i ovenstående udregning ændres til $d(x_i - x_j)/dx_j = -1$. Kort sagt $\mathbf{F}_j = -\mathbf{F}_i$. Hvilket ikke er overraskende, fordi udregningen ville være det matematiske bevis for Newtons tredie lov. Hvis partikel i påvirker partikel j med en kraft, så påvirker partikel j partikel j med en lige så stor men modsat rettet kraft.

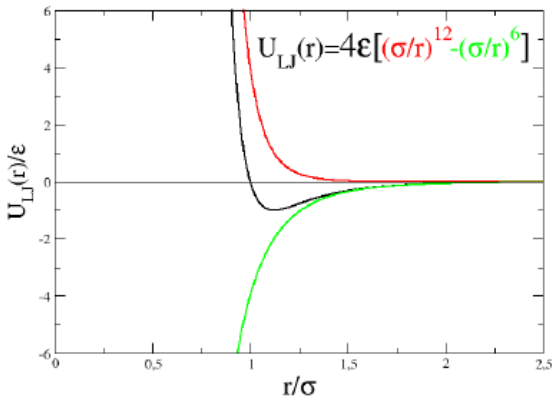
Rigtige molekyler har en mere kompliceret struktur end hvad vi kan beskrive alene med fjedrer. Hvis vi f.eks. har 3 kulstof atomer , der er bundet sammen med enkelt bindinger: C-C-C. Ligevægtsvinklen mellem den første og anden binding $\Theta_0 = 109.5\,\mathrm{deg}$. Men det kan vi bygge ind i en model ved at anvende en vinkelfjeder: $U(\Theta) = \frac{1}{2}k(\Theta - \Theta_0)^2$ hvor k er en fjeder konstant for bøjning. Vinklen Θ kan udtrykkes ved koordinaterne for de 3 kulstof atomer. Efter en lang udregning, kan vi finde ud af hvad for kræfter, sådan et potential giver ophav til. Hvis vi ville simulerer stive molekyler skal vi blot sætte $\Theta_0 = 180\,\mathrm{deg}$ og vælge en "stor"vinkelstivhedskonstant k.

Når vi vil modellere molekyler, så kan vi altså opfinde forskellige potentialer der beskriver strukturene, og differentierer disse for at udregne kræfterne som vi skal bruge i vores simulation når vi flytter rundt på partiklerne.

1.4 Intermolekylære vekselvirkninger



Figur 1.1: DPD potential



Figur 1.2: Lennard-Jones potentialet. Sort: hele udtrykket, rød: den repulsive del, grøn: den attraktive del.

Ovenstående har vi talt om kræfter inden for et molekyle, men hvad med kræfterne mellem forskellige molekyler? Hvis der er nogle vekselvirkninger, så er vores system en ideal gas. Vi har derfor brug for intramolekylre vekselvirkninger hvis vi vil modellere væsker eller faste stoffer.

Vi kan bygge mange forskellige modeller, men den simpleste er at antage at der er en repulsiv fjeder mellem partikler, der er ”tæt”på hinanden, men ingen fjeder hvis molekylerne blot er ”langt nok”fra hinanden. En sådan kraft kan skrives

$$F(r) = \begin{cases} F_{max}\left(1 - \frac{r}{\sigma}\right) & r < \sigma \\ 0 & r \geq \sigma \end{cases} \tag{1.12}$$

her er σ (græsk sigma) afstanden hvor fjederen holder op. Men for kortere afstande er der en repulsiv kraft mellem par af partikler, der er meget svag når $r \approx \sigma$ men vokser jo tættere partiklerne kommer på hinanden. Den maksimale repulsive kraft er F_{max} når $r \approx 0$. Denne type af væske modeller kaldes for Dissipative Particle Dynamics.[5] Vi tænker typisk på en DPD partikel som svarende til ca. 3 vand molekyler.

En mere realistisk model af vekselvirkningen mellem atomer tager udgangspunkt i kombinationen mellem attraktive van der Waals vekselvirkninger og stærke repulsive kræfter ved korte afstande når molekylers elektronskyer starter med at overlappe. Dette kan modelleres ved et Lennard-Jones (LJ) potential[6]

$$U(r) = 4\epsilon\left(\left(\frac{r}{\sigma}\right)^{12} - \left(\frac{r}{\sigma}\right)^{6}\right) \tag{1.13}$$

her er ϵ en energiskala, mens σ her er en længdeskala for hvor store atomerne er. Når $r \gg \sigma$ er potentialet 0 og partikler vekselvirker ikke. Potentialet har en attraktiv brønd af dybde $-\epsilon$ og er dybest ved afstanden $2^{1/6}\sigma$. For korte afstande $r < 0.95\sigma$ er potentialet stærkt repulsivt. For eksempel som model for Argon kan det f.eks. bruges at $\epsilon/k_b = 120K$ og $\sigma = 0.34nm$, der findes tilsvarende relationer for andre ædelgasser. Med LJ vekselvirkningen kan vi modellere hvordan en ædel gas ved lave temperaturer kondenserer til en væske, og ved endnu lavere temperaturer fryse til et krystal.

1.5 Tidsskridt

Når vi kører simulationer skal vi bestemme tidsskridtet Δt. Hvis det er "for stort"så er simulationen numerisk ustabil, dvs. simulationen eksploderer og alle partiklerne flyver væk i vilkårlig retninger. Hvis det er "for småt"så spilder vi computertiden og kommer ingen vegne. For en simulation af en fjeder kan vi komme med et estimat af hvad et fornuftigt tidsskridt er.

Hvis massen af to partikler er m og de er forbundet med en fjeder med fjederkonstant k så er frekvensen $f = (2\pi)^{-1}\sqrt{k/m}$ dvs. frekvensen bliver størrer når fjederen gøres stivere, eller hvis når massen gøres lettere. Perioden for svingningen er $P = 1/f$

$$P = \frac{2\pi\sqrt{m}}{\sqrt{k}} \tag{1.14}$$

[5] https://en.wikipedia.org/wiki/Dissipative_particle_dynamics
[6] https://en.wikipedia.org/wiki/Lennard-Jones_potential

For at simulere et system, så må tidsskridtet være lille målt i enheder af systemets korteste karakteristiske tid. For fjederen er det et tidsskridt, der er kort i forhold til perioden af svingningen, f.eks. $\Delta t = 0.01P$. Så bruger vi 100 tidsskridt på at opløse en svingningsperiode.

Dette betyder at hvis vi reducerer masser eller øger stivheden af fjedrer så skal vi bruge et kortere tidsskridt. Generelt så afhænger tidsskridtet af systemet, i en simulation af solsystemet kan man sagtens vælge tidskridt på flere timer eller dage. I en simulation af atomerne i et molekyle vil man typsk bruge tidsskrift på nogle få femtosekunder.

1.6 Temperatur

I klassisk mekanik har vi ikke nogen ligning for temperatur. Solsystemet har ikke en meningsfuld temperatur. Men for væsker af molekyler vil vi gerne lave simulationer ved en bestemt temperatur, så f.eks. kan se hvordan de krystalliserer ved lave temperaturer.

Temperatur er relateret til molekylernes bevægelse. Hvis vi forestiller os et system ved det absolutte nulpunkt, så står alle partiklerne stille. Hvis vi gerne vil varme det op, så skal vi få partiklerne til at bevæge sig. Det kan vi gøre ved at give dem en lille tilfældigt spark med en kraft $\mathbf{F}_{spark}$ i hver eneste tidsskridt.

De tilfældige spark tilfører varme til systemet, men problemet er at så vil energien vokse uden begrænsning. Vi har brug for en modsat rettet effekt, som kan tage varme ud af systemet. Det er netop det som friktion gør. Friktionskraften er $\mathbf{F} = -\gamma\mathbf{v}$, hvor $\mathbf{v}$ er hastigheden og γ er en friktionskoefficient. Friktionskræften er altid modsat hastigheden, således at den får partiklerne til at bevæge sig langsommere. Dvs. et system hvor der kun er friktion, så vil systemet ende med at stå stille, dvs. være i det absolutte nulpunkt.

En termostat styrer temperaturen, så nedenstående er Newtons anden lov med en termostat:

$$m\mathbf{a} = \mathbf{F} - \gamma\mathbf{v} + \mathbf{F}_{spark} \tag{1.15}$$

Den første kraft på højre hånd er den resulterende kraft fra partiklernes indbyrdes vekselvirkninger. De to sidste led er vores termostat friktion og tilfældige spark. Sådan en bevægelsesligning kaldes for en Langevin ligning.[7]

Temperaturen fremgår ikke direkte af bevælgelsesligningen, men er gemt i styrken af de tilfældige spark i forhold til friktionen. Vi kan måle temperaturen af en

[7] https://en.wikipedia.org/wiki/Langevin_equation

simulation ved direkte at kigge på den gennemsnitlige kinetiske energi af partiklerne:

$$\langle E_{kin} \rangle = \left\langle \frac{1}{2} m \mathbf{v}^2 \right\rangle = \frac{3 k_B T}{2} \tag{1.16}$$

Dette kaldes for equipartitionsteoremet.[8] Tretallet i tælleren kommer fordi den kinetiske energi har bidrag fra hastighedskomponenterne langs xyz. Som forventet så når partikler bevæger sig hurtigerer så er det fordi temperaturen er højere.

I DPD simulationer anvender vi et lidt mere kompliceret udtryk for tilfældige spark og friktion. Det har den fordel at DPD væsken opfylder Navier-Stokes ligningen for væsker.[9]

1.7 Enheder

I den virkelige verden har vi forskellge systemer af enheder, som f.eks. SI enhederne. Hvis vi ønsker at forstå solsystemet så er afstanden fra solen til Pluto 5×10^{12} meter mens afstanden mellem to kulstof atomer i et molekyle er 1.54×10^{-10} meter. Sådanne vanvittige tal gør det svært at forholde sig til hvad der foregår. Det peger på at SI enheder er meningsløse for disse systemer. Det er mere naturligt at vælge specifikke enheder, der er tilpasse de systemer vi ønsker at forstå.

Vælger vi i stedet som astronomisk enhed $1AU$ som værende middelafstanden fra solen til Jorden. Så er afstanden til Pluto $38.5AU$, så har vi en fornemmelse for hvor langt det er. Vælger vi som molekylær længdeskala f.eks. middelafstanden mellem elektronen og kernen i et Brint atom $1a_0 = 5.3 \times 10^{-11}$ meter (Bohr radius), så er en kulstof binding $2.9a_0$ lang. Med et naturligt valg af enheder så får vi tal værdier, som er meget letterer at forholde sig til.

I simulationerne vælger vi stort set altid at anvende sådanne naturlige enheder. Men i starten så virker disse enheder meget unaturlige og svært at forholde sig til.

I en simulation af en DPD væske dvs. partikler med repulsive fjedrer mellem sig, så vælger vi σ som længdeskala. Typiske partikel tætheder for disse simulationer er 3 partikler per kubik σ dvs. $\rho = 3\sigma^{-3}$. I en simulation af en LJ væske vælger vi stadig σ som længdeskala, typiske tætheder er så $\rho = 0.2 - 0.9\sigma^{-3}$. I SI enheder er $\sigma \approx 1nm$.

Typisk simulerer vi systemer hvor alle partikler har den samme masse m. Så er det naturligt at vælge netop m, som enheden for masse. Det har konsekvensen af partikler får massen $1m$ dvs. værdien 1 i enheder af m.

I SI enheder måler vi energi i Joule, mens temperatur måles i Kelvin, men det er et arbitrært valg der skyldes historiske tilfældigheder og hvornår vand koger og

[8] https://en.wikipedia.org/wiki/Equipartition_theorem

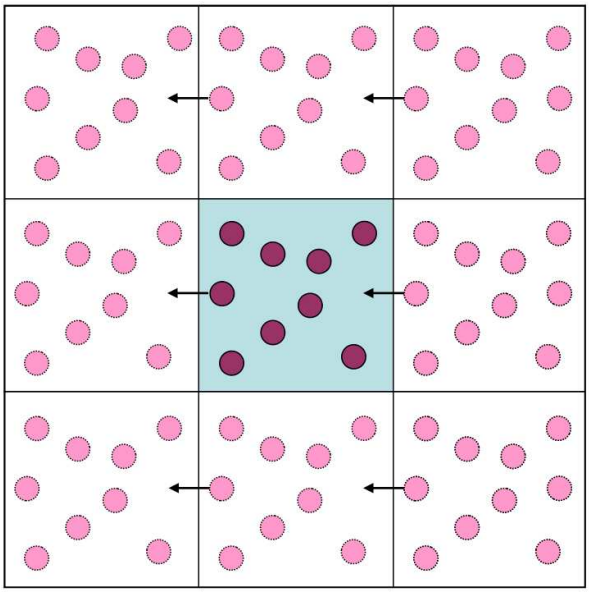
[9] https://en.wikipedia.org/wiki/Navier-Stokes_equations

fryser. Det bestemmer at Boltzmanns konstant k_B har en bestemt værdi. Da vi kan altid gange en temperatur med Boltzmanns konstant for at få en tilsvarende energi, så behøver vi ikke en selvstændig enhed for temperatur.

Som energiskala så er det typisk for de simulationer vi udfører, at vælge $k_B T_R$ hvor T_R er stue temperatur dvs. $300K$. Vi vælger også at angive temperatur i energienheder (dvs. vi definerer at $k_B = 1$). Konsekvensen er at en simulation med temperaturen $T = 1$ beskriver til hvad der sker ved stue temperatur. En simulation ved $T = 0.5$ svarer til halvdelen af stue temperatur ca. $150K$, det er altså relativt koldt. En simulation $T = 2$ svarer til ca. $600K$, det er altså relativt varmt. Med disse enheder så fryser vand ved $T = 270K/300K = 0.9$ og det koger ved $T = 370K/300K = 1.23$.

Bemærk at sålænge vi analyserer simulationsdata, så behøver vi ikke at konvertere vores værdier til SI enheder. Det mest naturlige er at bruge simulationsenheder. Det er først når vi sammenligner simulations data med eksperimentelle data, at vi skal udtrykke vores simulationsenheder i SI enheder.

1.8 Grænsebetingelser



Figur 1.3: Periodiske grænse betingelser

Når vi simulerer et system så kan vi typisk kun simulere relativt små systemer med de computere vi har til rådighed. I simulationen har vi en simulationskasse som inde-

holder hele vores system. Længden af simulationskassen er typisk 2-20 nanometer i SI enheder.

Hvad sker der når en partikel forsøger at forlade vores simulationskasse? Vi kan vælge at smide den væk, men så taber vi partikler. Vi kan gøre simulationskassen størrer, men så falder tætheden af systemet. Vi kunne også vælge at reflekterer partiklen tilbage ind i simulationskassen. Men det påvirker små systemer relativt meget, fordi en stor del af partiklerne er i nærheden af en væg.

Den mest blide grænsebetingelse er at antage periodiske grænser, dvs. når en partikel bevæger sig ud af simulationskassens venstre side, så stopper vi ind igen af simulationskassens højre side. Dette er illustreret på figuren.

1.9 Supercomputere

Hvis vi kører en MD simulation af N partikler, så kan vi antage at hver partikel vekselvirker med $z \ll N$ naboer. Det betyder at antallet af kraft beregninger vi skal lave i hver tidsskridt er $\sim Nz/2$. Den halve kommer fra Newtons 3 lov, har vi udregnet kraften fra j på i så ved vi allerede hvad kraften fra i på j er. Dvs. antallet af kraft beregninger er proportionalt med antal partikler i vores simulation.

En supercomputer er en masse hurtige computere med mange kerner forbundet med et meget hurtigt netværk (infiniband). MD simulationer har den fordel at de mange partikler kan fordeles ligeligt på forskellige kerner. Har vi f.eks. 256 kerner til rådighed, så kører simulationen ca. 256 gange hurtigerer. Det betyder at hvis det tager et år at køre simulationen på en enkelt kerne, så får vi resultatet efter halvanden dag på en supercomputer.

Kapitel 2

Praktisk Information

Til at udføre simulationsprojekterne skal i anvende simulationsprogrammet LAMMPS der er installeret på SDUs supercomputer Ucloud. Her introducerer vi LAMMPS og Ucloud.

2.1 LAMMPS

Large Atomic Molecular Massively Parallel Simulator forkortet LAMMPS[1] er en MD simulationskode, dvs. alle ligningerne fra teori sektionen ovenover er implementeret. LAMMPS er beregnet til at køre på supercomputere, og har ikke noget grafisk brugerinterface.

På supercomputere står simulationerne der skal køres i en kø, og når der er frie ressourcer så startes de automatisk. På den måde kører supercomputeren simulationer 24x7 på alle kerner, og når brugeren har sendt simulationen afsted til køen, så kan vi blot vente på at den bliver kørt.

En LAMMPS simulation styres af en kontrol fil (*.lam). Disse er tekstfiler, der kan læses og ændres med notepad. Til nogle simulationer skal LAMMPS også bruge en template for hvordan et molekyle ser ud. Disse filer hedder *.mol. Andre simulation kræver at vi har forberedt et helt system. Systemer er gemt i filer der hedder *.input. Filerne skal ligge i samme folder som .lam filen. Alle filerne i skal bruge kan I finde på github linket nedenfor.

En LAMMPS simulation består af følgende skridt

- Definer vekselvirkningerne for den model vi vil simulerer
- Indlæs eller skab en start konfiguration

[1] https://www.lammps.org

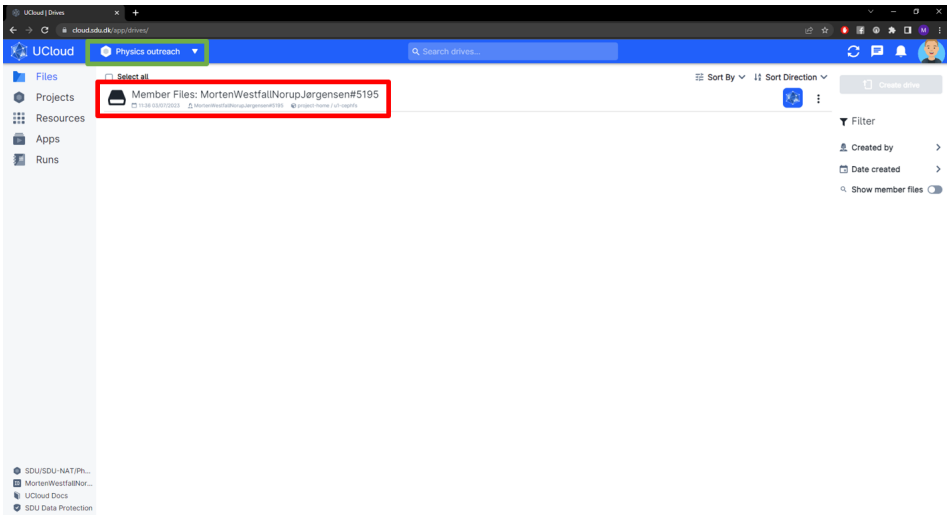
- Definer bevægelsesligningen for modellen
- Definer hvad for data vi vil gemme i løbet af simulationen
- Kør simulationen

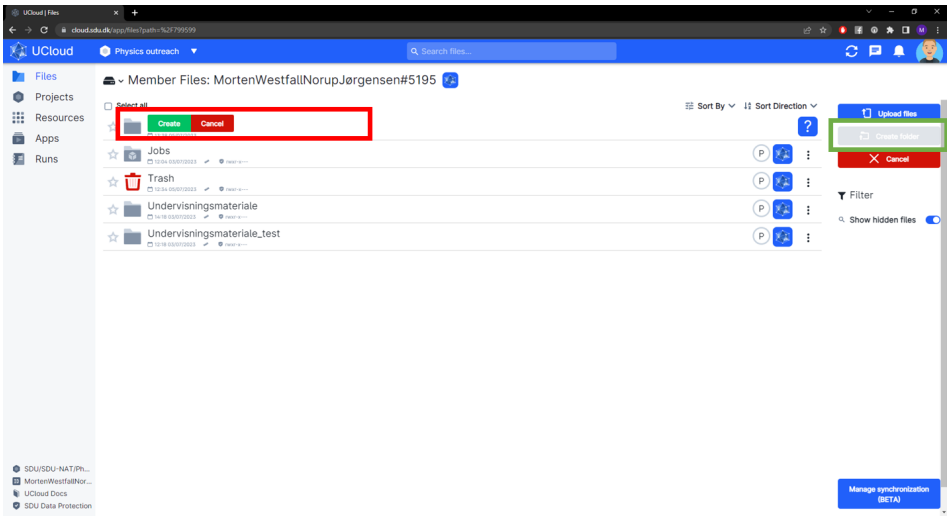
I løbet af simulationen danner LAMMPS en masse ekstra filer, evt. også billeder og en film af simulationen, således at I kan se hvad der er sket. Mens LAMMPS kører så udskriver den en masse information på skærmen. Denne skal I ikke bruge til noget. Men der kan i se f.eks. tidsskridt, temperatur, kinetisk og potentiel energi som simulationen kører.

2.2 Simulationer på Ucloud

Ucloud gør det muligt at køre forskellige software via din webbrowser. På den måde slipper du for selv at installere LAMMPS. Ucloud kan tilgås via `cloud.sdu.dk`. Herunder beskrives det hvordan du bruger Ucloud og kører simuleringer ved hjælp af Large Atomic Molecular Massively Parallel Simulation (LAMMPS) for at visualisere og undersøge soft-matter materialer.

1. Start med at logge in på Ucloud `https://cloud.sdu.dk/`
2. Du får udleveret brugernavn og password af underviseren på SDU.
3. Når du er logget ind på Ucloud kan du vælge hvilket projekt du skal arbejde fra (grøn firkant), vælg da ”Physics Outreach”. Dernæst kan du klikke dig in på projektet (rød firkant).
4. Dernæst kan du oprette en mappe (grøn firkant) hvori de simuleringer du køre kan gemmes og køres fra.
5. Hint: Husk at navngive mapperne (rød firkant) så det giver mening, og så du senere kan huske hvor du har gemt hvilke filer. Hvis du f.eks. skal køre flere simulationer med Brownsk bevægelse, kan man lave en overordnet mappe der hedder **Brownsk_bevægelse** og lave en undermappe for hver simulering man vil køre. Simulationerne danner en del filer, så man mister hurtigt overblikket hvis man kører flere simulationer i den samme folder.

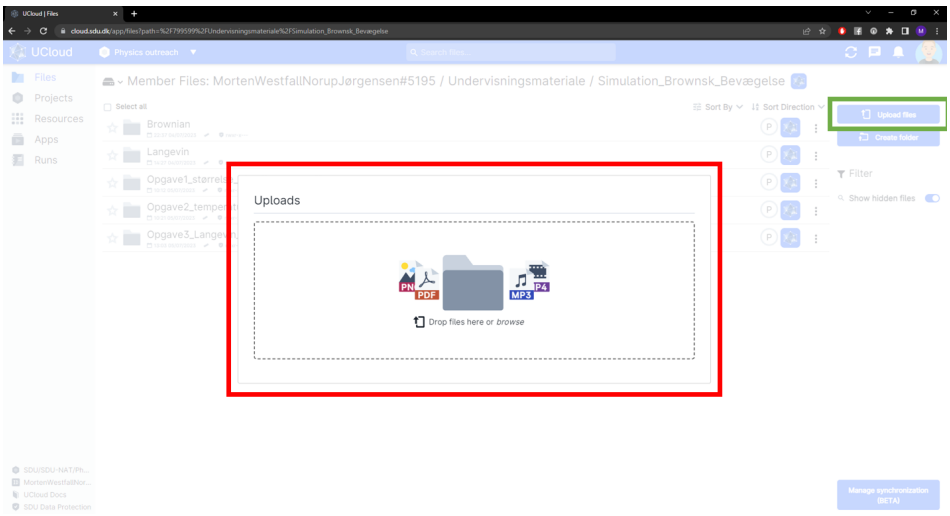

UCloud
Physics outreach
Search drives...
Files
Projects
Resources
Apps
Runs
Select all
Member Files: MortenWestfallNorupJørgensen#5195
Sort By
Sort Direction
Create drive
Filter
Created by
Date created
Show member files
SDU/SDU-NAT/Ph...
MortenWestfallNor...
UCloud Docs
SDU Data Protection



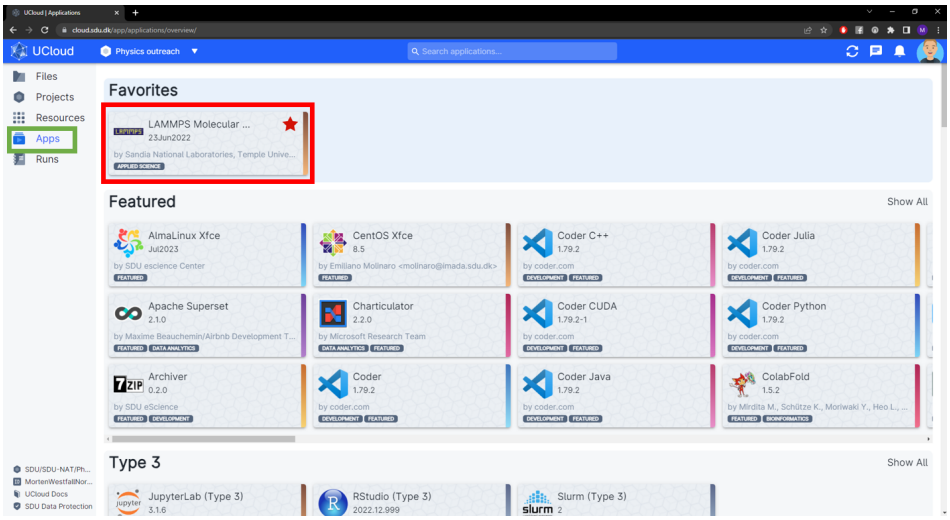
6. Simulationsfilerne du skal bruge kan du finde på `github.com/zqex/SDUPhysicsOutreach` Der er en folder for hver simulationsprojekt, og underfoldere for hver opgave i projektet.

7. Du downloader de filer du har brug for til din computer.

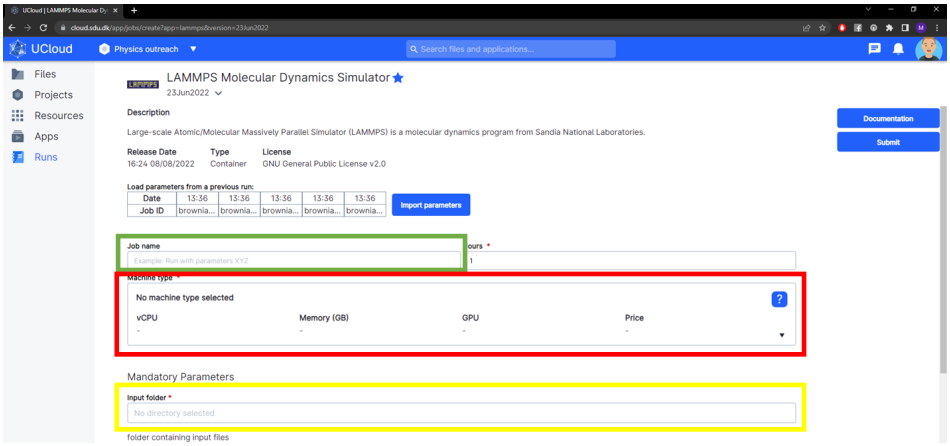
8. For at oploade filer til Ucloud mappen, skal du først gå ind i den mappe du vil lægge filerne op i. Dernæst skal du trykke på "Upload files"(grøn firkant) og trække/slippe filer ind i "Uploads"(rød firkant).



9. For at køre simuleringer skal du starte med at trykke på ”Apps”(grøn firkant). Dernæst skal du finde ”LAMMPS”(rød firkant). Hvis denne ikke kan ses direkte, kan man gå op og søge efter den i søgefeltet ”Search applications”. Når du har fundet LAMMPS trykker du på den.

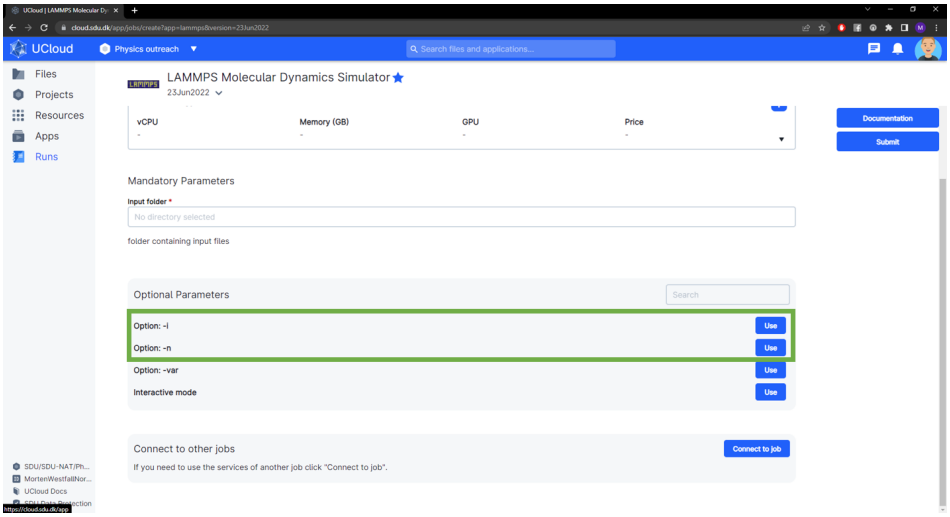


10. Du vil derefter komme til denne side som ses herunder. Det er dette sted du skal starte, hver gang du skal køre en ny simulering. Start med at give jobbet et navn (grøn firkant), det er igen en god idé at give jobbet et strategisk navn, f.eks. efter opgave og parametre. Under ”Machine type”(rød firkant) vælges ”u1-standard-16”. Dette vælger at køre simulationen på 16 kerner. Det er denne maskine vi ønsker at køre simuleringen fra. Derefter skal der vælges en ”Input folder”(gul firkant), det er i denne mappe alle output-filer kan findes. Når man trykker på ”No directory selected”kan man finde den mappe der skal bruges og trykke ”Use folder”.



11. Dernæst skal du vælge at bruge ”Option: -i”og ”Option: -n”ved at trykke på ”Use”ud for hver af disse to (grøn firkant). Når du har trykket ”Use”på dem

begge, skal du vælge den input-fil du ønsker at bruge til at køre simulationen. Her vil du skulle bruge den fil i inputmappen der slutter på ".lam". Når du har fundet denne fil kan du trykke "Use file". Ved "Number of processors"skriver du blot 16.



12. Her ses et eksempel på en simulation inden der trykkes "Submit". Når du har valgt de korrekte filer og skrevet de korrekte værdier ind, er du klar til at køre simuleringen ved at trykke "Submit".

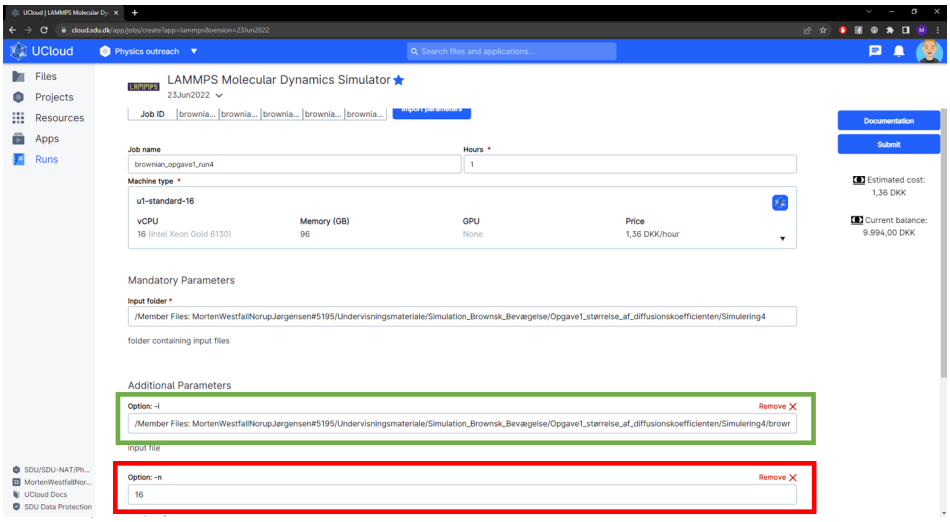
13. Simulationen står nu i køen på Ucloud og venter på at der bliver ressourcer ledige. Du kan submitte flere simulationer efter hinanden, og behøver ikke at vente på at den første bliver færdig.

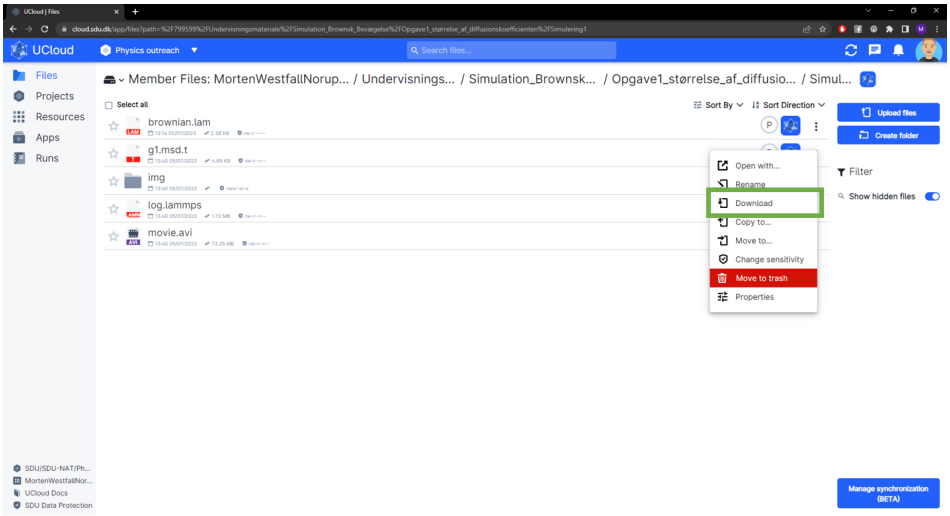
14. Det burde tage mindre end 5 minutter for at en simulation bliver færdig, når den først er begyndt.

15. Du kan se hvornår simuleringen er færdig, ved at trykke på "Runs"i venstre side og afvente til der står "Succes"ved simuleringen.

16. For at downloade output-filerne kan du igen finde den mappe du brugte som inputmappe, og trykke på de tre prikker ud fra filen og klikke download (grøn firkant).

17. Efter ca. en måned bliver filerne på UCloud projektet slettet, så husk at downloade alle de filer du skal bruge til rapporter oa. i god tid.


UCloud
Physics outreach
Files
Projects
Resources
Apps
Runs
LAMMPS Molecular Dynamics Simulator
23Jun2022
Job ID
Job name
brownian_opgave1_run4
Hours
1
Machine type
u1-standard-16
vCPU
16 (Intel Xeon Gold 6130)
Memory (GB)
96
GPU
None
Price
1,36 DKK/hour
Documentation
Submit
Estimated cost:
1,36 DKK
Current balance:
9.994,00 DKK
Mandatory Parameters
Input folder
/Member Files: MortenWestfallNorupJørgensen#5195/Undervisningsmateriale/Simulation_Brownsk_Bevægelse/Opgave1_størrelse_af_diffusionskoefficienten/Simulering4
folder containing input files
Additional Parameters
Option: -i
Remove
/Member Files: MortenWestfallNorupJørgensen#5195/Undervisningsmateriale/Simulation_Brownsk_Bevægelse/Opgave1_størrelse_af_diffusionskoefficienten/Simulering4/browr
input file
Option: -n
Remove
16


UCloud
Physics outreach
Files
Projects
Resources
Apps
Runs
Member Files: MortenWestfallNorup... / Undervisnings... / Simulation_Brownsk... / Opgave1_størrelse_af_diffusio... / Simul...
Select all
Sort By
Sort Direction
Upload files
Create folder
brownian.lam
g1.msd.t
img
log.lammps
movie.avi
Open with...
Rename
Download
Copy to...
Move to...
Change sensitivity
Move to trash
Properties
Filter
Show hidden files
Manage synchronization (BETA)

Kapitel 3

Gymnasieprojekt: Selv-bygning af surfaktanter

3.1 Formål

Visse molekyler kaldt surfaktanter[1] har evnen til spontant at danne komplicerede strukturer, dette princip kaldes selv-bygning[2]. Lipid molekyler[3] danner for eksempel spontant membraner. Vores celler er omkranset af sådanne lipid membraner. Det var formodeligt et meget tidligt og vigtigt skridt i livets udvikling at "ur-suppen"blev til individuelle organismer med en membran omkring sig.

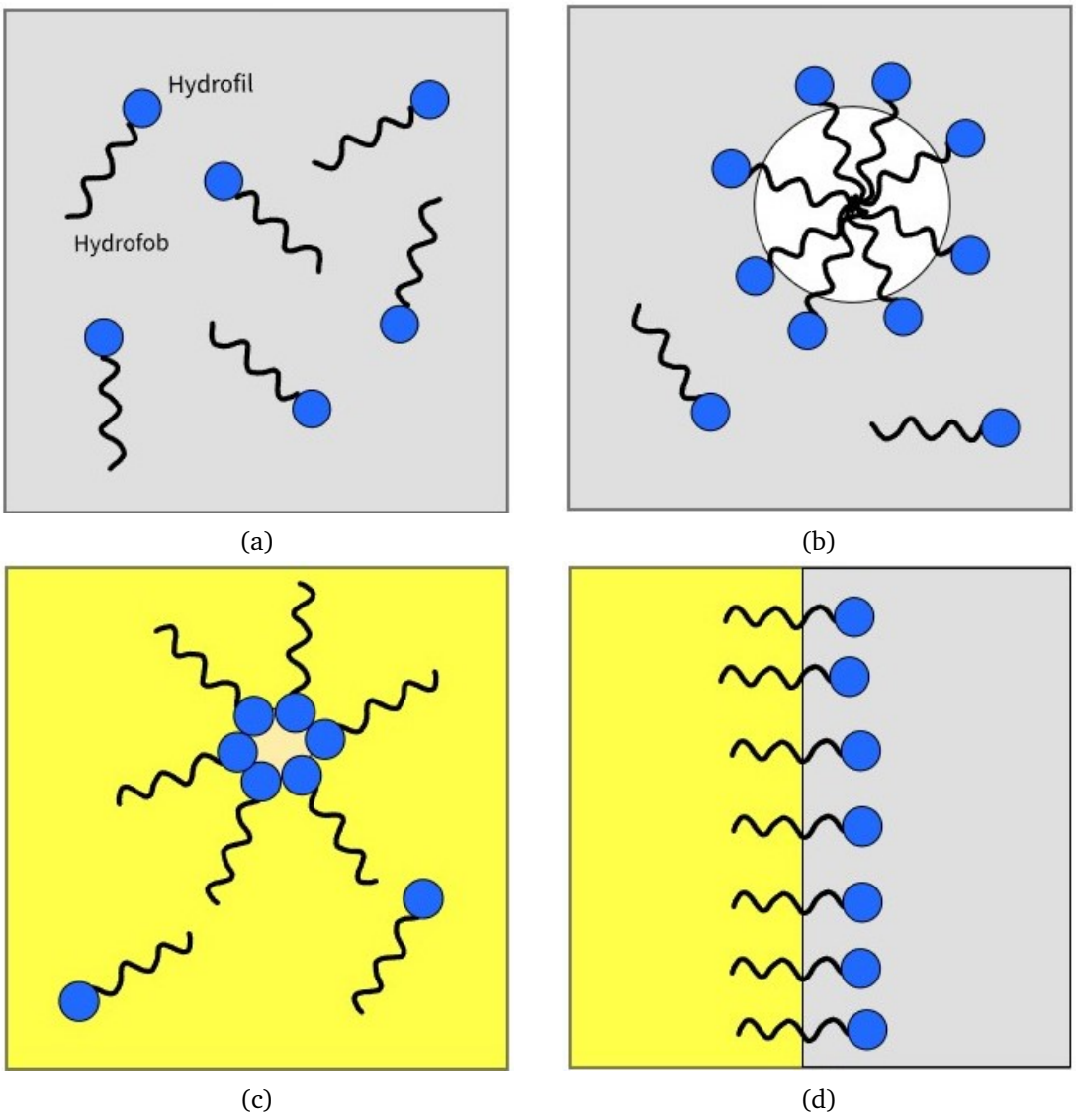
Formålet med denne øvelse er at studere dette selv-bygning fænomen ved at undersøge micelledannelse og emulsioner ved hjælp af computersimuleringer. Vi anvender DPD modeller til at beskrive blandinger af vand, olie og surfaktanter.

3.2 Teori

Surfaktanter er en vigtig klasse af kemiske forbindelser, der spiller en afgørende rolle i mange hverdagsprodukter og industrielle processer. En vigtig anvendelse af surfaktanter er i rengøringsmidler og vaskeprodukter. Deres evne til at sænke overfladespændingen gør dem i stand til at trænge ind i og løsne snavs og fedt fra overflader, så det lettere kan skylles væk. Surfaktanter fungerer også som emulgatorer, hvor man f.eks. i æggeblommer kan finde en naturlig forekommende surfaktant kaldet lecitin,

[1] https://en.wikipedia.org/wiki/Surfactant
[2] https://en.wikipedia.org/wiki/Self-assembly
[3] https://en.wikipedia.org/wiki/Lipid

som gør at vi kan lave mayonnaise.



Figur 3.1: (a) Surfaktanter opløst i vand. (b) Surfaktanter opløst i vand med micelledannelse. (c) Surfaktanter opløst i olie med micelledannelse. (d) Surfaktanter opløst i både olie og vand. Olien indkapsles i micellerne.

Surfaktanter består typisk af to dele: en hydrofil (vandelskende) hovedgruppe og en hydrofob (vandafvisende) hale, som vist i figur 3.1a. Den hydrofile del vil gerne interagere med vandmolekyler, mens den hydrofobe del helst vil undgå vandmolekyler

og derfor søger at minimere kontakten med det. Da hoved og hale sidder sammen må molekylerne som et hele finde på et kompromis.

På grund af denne opbygning med en hydrofil og hydrofob gruppe, kan surfaktanter opløses i vand og danne forskellige strukturer, som illustreret i figur 3.1b. Når surfaktanter opløses i vand, vil den hydrofile del orientere sig mod vandet, mens den hydrofobe del skærmer sig selv fra vandet. Denne molekylære sammensætning kaldes en micelle. Indeni micellekernen vil der være en lavere koncentration af vand end uden om micellen. På samme måde kan surfaktanter, når de opløses i olie, danne den modsatte struktur, hvor den hydrofobe del vender ud mod olien, og den hydrofile del forsøger at skærme sig selv inde i midten, se figur 3.1c.

Hvis vi har en opløsning med både vand og olie, som normalt ikke kan blandes, kan man tilføje surfaktanter. Derved vil surfaktanterne sætte sig i grænsen mellem olien og vandet, se fig. 3.1d. Hvis der er mere vand end olie vil surfaktanterne sætte sig på overfladen af olie dråber, det reducerer overfladespændingen, således at oliedråberne kan opløses i vand. Opvaskemiddel er netop surfaktanter og indkapsler olien, der gør det muligt for olien at blive opløst og skyllet væk mens vi vasker op.

3.3 Øvelsesvejledning

I denne øvelse skal I simulere forskellige sammensætninger af vand, olie og surfaktanter. Før I starter simuleringerne, er der nogle korte spørgsmål, I skal forsøge at besvare:

- Kan vand blandes med olie? Skitsér, hvordan det ville se ud, hvis man prøvede at blande det.
- Beskriv micelledannelsen, hvis surfaktanter opløses i vand eller olie. Skitsér, hvordan begge scenarier vil se ud.
- Hvis vi forestiller os, at der er meget få surfaktanter i en opløsning af vand og olie, hvordan vil disse surfaktanter så strukturere sig, og vil de have en virkning? Skitsér, hvordan I tror, det vil se ud.
- Hvad hvis der i stedet er mange surfaktanter i en opløsning af vand og olie?

Disse spørgsmål vil hjælpe jer med at forstå, hvad der sker under simuleringerne, og hvad I kan forvente at observere i de forskellige scenarier med vand, olie og surfaktanter.

For at køre simulationerne skal I finde mappen **Surfaktanter**. Heri ligger 6 mapper hvor der i hver mappe kan findes en inputfil med en **.lam** fil-endelse. Hver af de seks filer har en forskellig sammensætning af olie, vand og surfaktanter, se tabel 5.1. Det er vigtigt at I, i den mappe I laver på Ucloud og kører simulationen fra, også kopiere filen **surfactant.mol** over, da den er nødvendig for at køre simulationen.

Filnavn	Vand	Olie	Surfaktanter
Olie_surfaktant	0%	80%	20%
Olie_vand	50%	50%	0%
Olie_vand_lidt_surfaktant	47%	47%	6%
Olie_vand_meget_surfaktant	30%	30%	40%
Vand_olie_middel_surfaktant	40%	40%	20%
Vand_surfaktant	80%	0%	20%

Tabel 3.1

Kør da alle 6 simuleringer og download videoerne (filerne med **.avi**-endelsen) fra hver simulation. I videoerne er vand hvidt, olie er gult, surfaktanter er grønne-blå.

Sammenlign resultaterne fra simulationsvideoerne med de skitser i tegnede tidligere - stemmer det overens, eller blev I overrasket? Hvilke del af surfaktanterne vil I identificere som den hydrofobe og hydrofile? Brug teorien til af beskrive fysikken i hver simulation.

3.4 Forslag til eksperimenter

Surfaktanter spiller som sagt en stor rolle i vores hverdag, derfor kan man let lave nogle enkle eksperimenter, der kan vise virkningen af surfaktanter.

Til dette eksperiment skal der bruges: olie, vand, opvaskemiddel og et stort glas.

- Start med at hælde vand og olie i glasset og se hvad der sker.
- Tilsæt nu gradvist mere og mere opvaskemiddel under omrøring og observér hvad der sker.
- Hvad sker der hvis I lader glasset hvile et døgn?

Overfladespænding kan vises ved at hælde vand i en tallerken. Drys f.eks. peber på overfladen. Vand har en relativt høj overfladespænding. Dryb en dråbe af sæbe på tallerkenen. Surfaktanterne (sæben) reducerer overfladespændingen der hvor dråben

rammer. Overfladen kan altså minimerer sin energi ved at lade sæben dække overfladen, men dette skubber peber kornene væk fra der hvor dråben landede. Sådanne strømninger skabt af forskelle i overfladespænding hedder Marangoni effekter[4].

[4]https://en.wikipedia.org/wiki/Marangoni_effect